

# **Dynamic Systems**

---

## **an introduction**

Rico A. R. Picone  
Department of Mechanical Engineering  
Saint Martin's University

17 April 2024

Copyright © 2024 Rico A. R. Picone All Rights Reserved

<b>I</b>	<b>Electromechanical system modeling</b>	<b>14</b>
<b>01 intro</b>	<b>Introduction</b>	<b>15</b>
01.1 intro.it	The systems approach . . . . .	24
01.2 intro.sdet	State-determined systems . . . . .	26
01.3 intro.lump	Energy, power, and lumping . . . . .	29
	Lumping . . . . .	29
01.4 intro.mecht	Mechanical translational elements . . . . .	32
	Translational springs . . . . .	33
	Point-masses . . . . .	34
	Dampers . . . . .	35
	Force and velocity sources . . . . .	35
01.5 intro.mechr	Mechanical rotational elements . . . . .	37
	Rotational springs . . . . .	38
	Moments of inertia . . . . .	39
	Rotational dampers . . . . .	40
	Torque and angular velocity sources . . . . .	40
01.6 intro.ele	Electronic elements . . . . .	42
	Capacitors . . . . .	42
	Inductors . . . . .	43
	Resistors . . . . .	44
	Sources . . . . .	45
01.7 intro.genvars	Generalized through- and across-variables	47
01.8 intro.genels	Generalized one-port elements . . . . .	49

<i>CONTENTS</i>		3
	A-type energy storage elements . . . . .	49
	T-type energy storage elements . . . . .	49
	D-type energy dissipative elements . . . . .	50
	Sources . . . . .	50
01.9 intro.exe	Exercises for Chapter 01 intro . . . . .	52
	Exe. intro.madrid . . . . .	52
<b>02 graphs</b>	<b>Linear graph models</b>	<b>53</b>
02.1 graphs.intro	Introduction to linear graphs . . . . .	54
02.2 graphs.sign	Sign convention . . . . .	57
	Electronic systems . . . . .	57
	Translational mechanical systems . . . . .	59
	Rotational mechanical systems . . . . .	61
02.3 graphs.connect	Element interconnection laws . . . . .	65
02.4 graphs.sysmod	Systematic linear graph modeling . . . . .	68
02.5 graphs.exe	Exercises for Chapter 02 graphs . . . . .	71
	Exe. graphs.playmate . . . . .	71
	Exe. graphs.nod . . . . .	71
	Exe. graphs.johnnycash . . . . .	72
	Exe. graphs.lillimoomie . . . . .	74
	Exe. graphs.varieties . . . . .	75
	Exe. graphs.cormac . . . . .	76
	Exe. graphs.kurt . . . . .	76
	Exe. graphs.bunker . . . . .	77
<b>03 ss</b>	<b>State-space models</b>	<b>79</b>
03.1 ss.svar	State variable system representation . . . . .	80
03.2 ss.ssmodel	State and output equations . . . . .	82
03.3 ss.graph2nt	Normal trees . . . . .	84
03.4 ss.nt2ss	Normal tree to state-space . . . . .	87
03.5 ss.trans	State-space model of a translational mechanical system . . . . .	91
03.6 ss.rot	State-space model of a rotational mechanical system . . . . .	95

03.7 ss.ss2tf2io	Bridge between state-space and io differential equations . . . . .	99
	Transfer functions . . . . .	99
	Bridging transfer functions and io differential equations . . . . .	99
	Bridging transfer functions and state-space models . . . . .	101
03.8 ss.exe	Exercises for Chapter 03 ss . . . . .	105
	Exe. ss.metroid . . . . .	105
	Exe. ss.megaman . . . . .	105
	Exe. ss.sonic . . . . .	106
	Exe. ss.nintendo . . . . .	107
	Exe. ss.supernintendo . . . . .	107
	Exe. ss.gameboy . . . . .	108
	Exe. ss.blowhard . . . . .	108
	Exe. ss.blinken . . . . .	109
	Exe. ss.chunker . . . . .	110
	Exe. ss.stevenuniverse . . . . .	110
	Exe. ss.winken . . . . .	111
	Exe. ss.granada . . . . .	112
	Exe. ss.valencia . . . . .	112
	Exe. ss.stevenash . . . . .	113
<b>04 emech</b>	<b>Electromechanical systems</b>	<b>115</b>
04.1 emech.trans	Ideal transducers . . . . .	116
04.2 emech.transmod	Modeling with transducers . . . . .	119
	State-space modeling with transducers . . . . .	119
04.3 emech.dcm	DC motors . . . . .	121
	Lorentz force . . . . .	121
	Permanent magnet DC motors . . . . .	122
	Wound stator DC motors . . . . .	124
	Brushless DC motors . . . . .	125
	A PMDC motor model . . . . .	126
	Motor constants . . . . .	126

	Animations . . . . .	127
04.4 emech.real	Modeling a real electromechanical system	128
	Linear graph model . . . . .	129
	State-space model . . . . .	130
04.5 emech.curves	DC motor performance in steady-state . .	133
	Modeling the test system . . . . .	133
	Steady-state performance analysis . . . .	134
04.6 emech.dcmtran	Transient DC motor performance . . . . .	140
	Simulating the step response . . . . .	141
	Estimating parameters from the step response . . . . .	144
04.7 emech.drive	Driving motors . . . . .	150
	Motor curves . . . . .	152
04.8 emech.exe	Exercises for Chapter 04 emech . . . . .	153
	Exe. emech.triangle . . . . .	153
	Exe. emech.square . . . . .	153
	Exe. emech.rectangle . . . . .	154
	Exe. emech.quadrilateral . . . . .	154
	Exe. emech.mrpotatohead . . . . .	155
	Exe. emech.clunker . . . . .	156
	Exe. emech.curvy . . . . .	157
	Exe. emech.chair . . . . .	157
	Exe. emech.onomatopoeia . . . . .	158
	Exe. emech.deglazification . . . . .	159
	Exe. emech.confuzzled . . . . .	159
	Exe. emech.levitation . . . . .	159

## II Time response 161

05 lti	<b>Linear time-invariant system properties</b>	<b>162</b>
05.1 lti.super+	Superposition, derivative, and integral properties . . . . .	163
05.2 lti.equistab	Equilibrium and stability properties . . .	165
	Stability defined by the free response . . .	166

	Stability defined by the forced response . . . . .	167
05.3 lti.vib	Vibration isolation table analysis . . . . .	168
	Linear graph and state-space models . . . . .	169
	Equilibrium . . . . .	170
	Transfer function model . . . . .	170
	Input-output differential equation . . . . .	171
	Step response . . . . .	171
	Stability . . . . .	175
05.4 lti.ghost	When gravity ghosts you . . . . .	176
05.5 lti.exe	Exercises for Chapter 05 lti . . . . .	180
	Exe. lti.oil . . . . .	180
	Exe. lti.water . . . . .	180
	Exe. lti.timmychalamet . . . . .	180
	Exe. lti.flopugh . . . . .	180
<b>06 trans</b>	<b>Qualities of transient response</b>	<b>182</b>
06.1 trans.char	Characteristic transient responses . . . . .	183
06.2 trans.firsto	First-order systems in transient response . . . . .	185
	Free response . . . . .	185
	Step response . . . . .	186
	Impulse and ramp responses . . . . .	186
06.3 trans.secondo	Second-order systems in transient response . . . . .	189
	Free response . . . . .	189
	Step response . . . . .	192
	Impulse and ramp responses . . . . .	193
	An example with superposition . . . . .	193
06.4 trans.exe	Exercises for Chapter 06 trans . . . . .	197
	Exe. trans.truman . . . . .	197
	Exe. trans.mogul . . . . .	197
	Exe. trans.kibble . . . . .	198
	Exe. trans.biology . . . . .	198
<b>07 ssresp</b>	<b>State-space response</b>	<b>199</b>
07.1 ssresp.respons	Solving for the state-space response . . . . .	201

	State response . . . . .	201
	State transition matrix . . . . .	202
	Output response . . . . .	204
07.2 ssresp.eig	Linear algebraic eigenproblem . . . . .	205
	Solving for eigenvalues . . . . .	205
	Solving for eigenvectors . . . . .	206
07.3 ssresp.eigcomp	Computing eigendecompositions . . . . .	208
	Matlab eigendecompositions . . . . .	208
	Python eigendecompositions . . . . .	210
07.4 ssresp.diag	Diagonalizing basis . . . . .	213
	Changing basis in the state equation . . . . .	213
	Modal and eigenvalue matrices . . . . .	213
	Diagonalization of the state equation . . . . .	214
	Computing the state transition matrix . . . . .	215
07.5 ssresp.vibe	A vibration example with two modes . . . . .	219
	Setting up the problem . . . . .	220
	Without damping . . . . .	221
	With a little damping . . . . .	225
07.6 ssresp.mixed	Analytic and numerical output response example in Matlab . . . . .	230
	Analytic solution . . . . .	231
	Numerical solution . . . . .	232
07.7 ssresp.sim	Simulating state-space response . . . . .	235
	Analytic solution . . . . .	236
	Numerical solution . . . . .	240
07.8 ssresp.exe	Exercises for Chapter 07 ssresp . . . . .	243
	Exe. ssresp.larry . . . . .	243
	Exe. ssresp.mo . . . . .	243
	Exe. ssresp.curly . . . . .	244
	Exe. ssresp.lonely . . . . .	244
	Exe. ssresp.artemis . . . . .	245
	Exe. ssresp.level . . . . .	246

<b>III Modeling other systems</b>	<b>248</b>
<b>08 thermoflumped-parameter modeling fluid and thermal systems</b>	<b>249</b>
08.1 thermoflu.flu Fluid system elements . . . . .	250
Fluid inertances . . . . .	251
Fluid capacitors . . . . .	253
Fluid resistors . . . . .	253
Flowrate and pressure drop sources . . . . .	254
Generalized element and variable types . . . . .	254
08.2 thermoflu.therm Thermal system elements . . . . .	256
Thermal capacitors . . . . .	257
Thermal resistors . . . . .	257
Heat flow rate and temperature sources . . . . .	259
Generalized element and variable types . . . . .	259
08.3 thermoflu.flutrans Fluid transducers . . . . .	261
08.4 thermoflu.dam State-space model of a hydroelectric dam . . . . .	263
Normal tree, order, and variables . . . . .	263
Elemental equations . . . . .	263
Continuity and compatibility equations . . . . .	264
State equation . . . . .	265
08.5 thermoflu.fem Thermal finite element model . . . . .	267
08.6 thermoflu.exe Exercises for Chapter 08 thermoflu . . . . .	274
Exe. thermoflu.tinker . . . . .	274
Exe. thermoflu.tailor . . . . .	274
Exe. thermoflu.soldier . . . . .	274
Exe. thermoflu.tpain . . . . .	274
Exe. thermoflu.dramp . . . . .	276
Exe. thermoflu.up . . . . .	277
<b>IV Fourier analysis</b>	<b>279</b>
<b>09 four Fourier series and transforms</b>	<b>280</b>
09.1 four.series Fourier series . . . . .	281
09.2 four.fsexample Complex Fourier series example . . . . .	285



	Part a: complex Fourier analysis . . . . .	286
	Part b: harmonic amplitude and phase with spectra . . . . .	289
	Part c: conversion to trig form . . . . .	292
09.3 four.transform	Fourier transform . . . . .	294
09.4 four.dft	Discrete and fast Fourier transforms . . .	303
	Leakage . . . . .	308
09.5 four.exe	Exercises for Chapter 09 four . . . . .	310
	Exe. four.stanislaw . . . . .	310
	Exe. four.pug . . . . .	310
	Exe. four.ponyo . . . . .	310
	Exe. four.seesaw . . . . .	310
	Exe. four.totoro . . . . .	310
	Exe. four.mall . . . . .	311
	Exe. four.miyazaki . . . . .	311
	Exe. four.haku . . . . .	312
	Exe. four.secrets . . . . .	312
	Exe. four.society . . . . .	315
	Exe. four.flapper . . . . .	315
	Exe. four.eastegg . . . . .	316
	Exe. four.savage . . . . .	316
	Exe. four.strawman . . . . .	317
<b>10 freq</b>	<b>Frequency response</b>	<b>318</b>
10.1 freq.fir	Frequency and impulse response . . . . .	319
	Frequency response functions . . . . .	319
	Frequency response . . . . .	320
	Impulse response . . . . .	321
10.2 freq.sin	Sinusoidal input, frequency response . . .	327
10.3 freq.bode	Bode plots . . . . .	330
10.4 freq.bodesimp	Bode plots for simple transfer functions .	332
10.5 freq.bodesketch	Sketching Bode plots . . . . .	335
10.6 freq.per	Periodic input, frequency response . . . .	339
10.7 freq.exe	Exercises for Chapter 10 freq . . . . .	344

Exe. freq.gauche . . . . . 344  
 Exe. freq.tickle . . . . . 344  
 Exe. freq.me . . . . . 345  
 Exe. freq.elmo . . . . . 346  
 Exe. freq.hum . . . . . 346

**V Laplace analysis 349**

**11 lap Laplace transforms 350**

11.1 lap.in Introduction . . . . . 351  
 11.2 lap.def Laplace transform and its inverse . . . . . 353  
     The Laplace transform . . . . . 353  
     The inverse Laplace transform . . . . . 354  
 11.3 lap.pr Properties of the Laplace transform . . . . . 358  
     Existence . . . . . 358  
     Linearity . . . . . 358  
     Time-shifting . . . . . 358  
     Time-differentiation . . . . . 359  
     Time-integration . . . . . 359  
     Convolution . . . . . 359  
     Final value theorem . . . . . 359  
 11.4 lap.inv Inverse Laplace transforming . . . . . 361  
     Inverse transform with a partial fraction  
     expansion in Matlab . . . . . 361  
     Just clubbing it with Matlab . . . . . 365  
 11.5 lap.sol Solving io ODEs with Laplace . . . . . 367  
 11.6 lap.exe Exercises for Chapter 11 lap . . . . . 370

**12 tf Transfer functions 371**

12.1 tf.zp Poles and zeros . . . . . 372  
     Pole-zero plots and stability . . . . . 373  
     Second-order systems . . . . . 374  
 12.2 tf.tformat Exploring transfer functions in Matlab . . 377  
     The tf command and its friends . . . . . 377

	Algebraic operations with <code>tf</code> s . . . . .	378
	State-space models to <code>tf</code> models. . . . .	378
	Poles, zeros, and stability . . . . .	380
	Simulating with <code>tf</code> s . . . . .	380
12.3 <code>tf.zpk</code>	ZPK transfer functions in Matlab . . . . .	383
12.4 <code>tf.exe</code>	Exercises for Chapter 12 <code>tf</code> . . . . .	385
	Exe. <code>tf.sallywag</code> . . . . .	385
	Exe. <code>tf.swashbuckling</code> . . . . .	385
	Exe. <code>tf.boris</code> . . . . .	385
<b>13 <code>imp</code></b>	<b>Impedance-based modeling</b>	<b>387</b>
13.1 <code>imp.ip</code>	Input impedance and admittance . . . . .	388
	Impedance of ideal passive elements . . . . .	389
	Impedance of interconnected elements . . . . .	390
13.2 <code>imp.2port</code>	Impedance with two-port elements . . . . .	392
13.3 <code>imp.tf</code>	Transfer functions via impedance . . . . .	395
13.4 <code>imp.examat</code>	Impedance modeling example in Matlab . . . . .	399
13.5 <code>imp.eqiv</code>	Norton and Thévenin theorems . . . . .	404
	Norton's theorem . . . . .	404
	Converting between Thévenin and Norton equivalents . . . . .	405
13.6 <code>imp.divide</code>	The divider method . . . . .	407
	Across-variable dividers . . . . .	407
	Through-variable dividers . . . . .	408
	Transfer functions using dividers . . . . .	408
13.7 <code>imp.exe</code>	Exercises for Chapter 13 <code>imp</code> . . . . .	411
	Exe. <code>imp.tile</code> . . . . .	411
	Exe. <code>imp.granite</code> . . . . .	411
	Exe. <code>imp.granted</code> . . . . .	411
	Exe. <code>imp.concrete</code> . . . . .	412
	Exe. <code>imp.rhine</code> . . . . .	412
	Exe. <code>imp.tableau</code> . . . . .	413
	Exe. <code>imp.gypsum</code> . . . . .	415

CONTENTS	12
<b>VI Nonlinear system analysis</b>	<b>416</b>
<b>14 nlin Nonlinear systems and linearization</b>	<b>417</b>
	Autonomous and nonautonomous systems 417
	Equilibrium . . . . . 418
14.1 nlin.lin	Linearization . . . . . 419
	Taylor series expansion . . . . . 419
14.2 nlin.char	Nonlinear system characteristics . . . . . 421
	Those in-common with linear systems . . 421
	Stability . . . . . 421
	Qualities of equilibria . . . . . 422
14.3 nlin.exe	Exercises for Chapter 14 nlin . . . . . 424
	Exe. nlin.sigmund . . . . . 424
	Exe. nlin.franz . . . . . 424
<b>15 phase Phase space analysis</b>	<b>426</b>
<b>16 sim Simulating Nonlinear Systems</b>	<b>427</b>
16.1 sim.python	Nonlinear Systems in Python . . . . . 428
	Defining a Nonlinear System . . . . . 428
16.2 sim.matlab	Nonlinear systems in Matlab . . . . . 432
	Defining a nonlinear system . . . . . 432
	Simulating a nonlinear system . . . . . 432
	Plotting the response . . . . . 434
16.3 sim.fluid	Nonlinear fluid system example . . . . . 436
	Normal tree, order, and variables . . . . . 436
	Elemental, continuity, and compatibility equations . . . . . 436
	State equation . . . . . 439
	Simulation . . . . . 440
	Nonlinear resistance . . . . . 441
16.4 sim.exe	Exercises for Chapter 16 sim . . . . . 445
	Exe. sim.freud . . . . . 445
	Exe. sim.kafka . . . . . 445
	Exe. sim.hootenanny . . . . . 446

<i>CONTENTS</i>		13
<b>VII Appendices</b>		<b>450</b>
<b>A</b>	<b>Mathematics reference</b>	<b>451</b>
01.1 math.quad	Quadratic forms . . . . .	452
	Completing the square . . . . .	452
01.2 math.trig	Trigonometry . . . . .	453
	Triangle identities . . . . .	453
	Reciprocal identities . . . . .	453
	Pythagorean identities . . . . .	453
	Co-function identities . . . . .	454
	Even-odd identities . . . . .	454
	Sum-difference formulas (AM or lock-in)	454
	Double angle formulas . . . . .	454
	Power-reducing or half-angle formulas .	455
	Sum-to-product formulas . . . . .	455
	Product-to-sum formulas . . . . .	455
	Two-to-one or harmonic amplitude formulas . . . . .	455
01.3 math.matrix	Matrix inverses . . . . .	456
01.4 math.lap	Laplace transforms . . . . .	457
<b>B</b>	<b>Advanced topics</b>	<b>458</b>
02.1 adv.eig	Systems with repeated eigenvalues . . . .	459
<b>C</b>	<b>Summaries</b>	<b>461</b>
03.1 sum.sysrep	Summary of system representations . . .	462
03.2 sum.els	Summary of one-port elements . . . . .	463
03.3 sum.lap	Laplace transforms . . . . .	464
03.4 sum.ft	Fourier transforms . . . . .	466
<b>D</b>	<b>Complex analysis</b>	<b>469</b>
04.1 com.euler	Euler's formulas . . . . .	470
<b>Bibliography</b>		<b>471</b>

# **Part I**

## **Electromechanical system modeling**

## 01 intro

**1 System dynamics** is the field that studies **dynamic systems**. And dynamic systems are those that change. But on a long-enough time scale, it's hard to find systems that *don't* change. If we are to be a bit more modest in our definition, although modest we are not, we can conclude: dynamic systems are those that change significantly on interesting time-scales. However, there is a further qualification in order, since a common thread runs through the entire field: that of **mathematical representation**. Every dynamic system studied has or could have a mathematical representation. "Mathematical representations" are here understood broadly, encompassing equations and also graphical depictions with implicit mathematical relations. With this in mind, we arrive at our final definition.

### Definition 01 intro.1: dynamic system

A dynamic system is one that changes significantly on interesting time-scales and can be represented mathematically.

This is actually quite broad. For instance, conceivably we could derive a mathematical model for the number of poodles barking in France or the sum of the diameters of all human eyes.

**2** With the relatively intuitive graphical forms of dynamic system representation described below, one might wonder why more explicitly mathematical representations are required at all. The answer is that while the design of dynamic systems is aided by such graphical representations, thorough mathematical analysis is indispensable for good design. With

such analysis we can predict, for instance, the maximum acceleration a person in a vehicle would experience under certain operating conditions.

3 Dynamic system mathematical models will frequently include the following types variables, which will be described in more detail later in this chapter.

**time** An independent time variable, often given the symbol  $t$ .

**parameters** Parameters are variables, usually considered constant, that describe the system's physical qualities (e.g. mass, length, electrical capacitance, thermal resistance, etc.).

**input variables** Input variables represent generally variable quantities independent of the system (e.g. external force, voltage source, pump pressure, etc.). These are usually given variants of the symbol  $u$ .

**output variables** Output variables are dependent variables that represent quantities of interest. (e.g. velocity of a vehicle, voltage across certain terminals, number of poodles barking, etc.). These are usually given variants of the symbol  $y$ .

**state variables** State variables are a minimal (but not unique) set of dependent variables that represent the internal status or "state" of the system (e.g. force through a spring, velocity of a mass, voltage of a capacitor, current through an inductor, etc.). These are usually given variants of the symbol  $x$ .

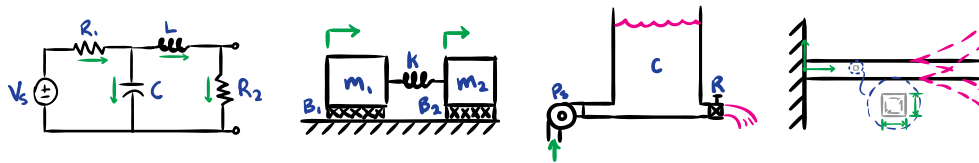
4 We here give an overview of some of the most important system representations, all of which are described in detail later in the text. It helps to get a lay of the land before we go exploring.

## Graphical representations

5 Several types of graphical representations can be useful. We begin with the venerable **diagram** or **schematic**, which should include all important elements of the system, such as those shown in [Fig. intro.1](#).

6 The mathematical relations in schematics are frequently rather implicit. For instance, the schematics of the two-mass system in [Fig. intro.1](#) explicitly specifies certain equations, such that the force applied by the spring  $k$  is





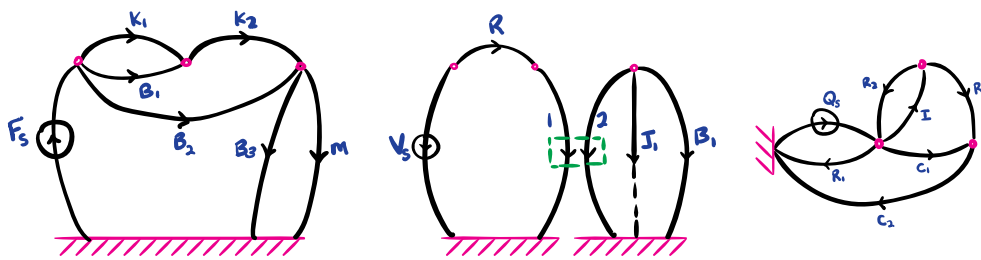
**Figure intro.1:** schematics representing (left-to-right) electronic, discrete mechanical, fluid, and continuous mechanical systems.

equal in magnitude and opposite in direction for each mass element  $m_1$  and  $m_2$ . The equations for each element, later called the “elemental equations,” are rarely explicitly given in any of the graphical system representations. In our two-mass example, for instance, each mass element would have Newton’s second law as its elemental equation. This is implied by the fact that it is represented as a mass. However, sometimes additional information is required. In the case of the spring  $k$  connecting the masses, it is reasonable to expect that the force in the spring is monotonically related to its length—this is (as we will see) what it means for an element to behave as a spring. However, from the schematic alone, it would be risky to assume the spring follows Hooke’s law—that is, that its force is *proportional* to its length. So many diagrams require a narrative supplement.

7 A more-precise and minimal graphical representation of a dynamic system is the **linear graph**, the subject of [Chapter 02 graphs](#).<sup>1</sup> Some sample linear graphs are shown in [Fig. intro.2](#).

8 Linear graphs are more-precise than schematics in that they explicitly connect elements (graph edges) at graph nodes. Explicit here are the structural relations among elements, which can be transcribed into equations. For instance, the electronic subsystem of the middle linear graph above tells us, *à la* Kierchhoff’s voltage law, that the voltage across each of the elements sum to zero:  $V_S - v_1 - v_R = 0$ . Similarly, *à la* Kierchhoff’s current law, the currents through each of the elements at the node connecting elements R and 1 sum to zero:  $i_R - i_1 = 0$ . We generalize these

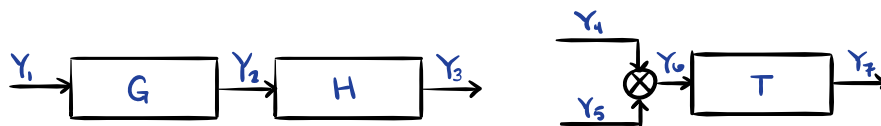
<sup>1</sup>It is worth mentioning a related type of graphical representation, the *bond graph*. The linear graph and bond graph representations are equivalent in many ways, but we prefer the linear graph for its intuitiveness.



**Figure intro.2:** linear graph representations of three systems. Elements are in black and nodes are in magenta.

relations for linear graphs in [Lec. 02.3 graphs.connect](#) Once again, the elemental equations are implicit.

9 The final type of graphical system representation we consider extensively in the text is the **block diagram**, a couple examples of which are shown in [Fig. intro.3](#).



**Figure intro.3:** block diagrams of two systems.

10 Block diagrams are more high-level than schematics and linear graphs, and usually show the interconnection of multiple dynamic systems. In [Fig. intro.3](#), the systems represented by blocks G and H are concatenated such that the output of G is the input of H. The contents of the blocks are systems usually interpreted in an explicitly mathematical form that will be introduced in a moment called the *transfer function* that relates the input and output variables; in our example, G maps  $Y_1$  to  $Y_2$ , H maps  $Y_2$  to  $Y_3$ , and T maps  $Y_6$  to  $Y_7$ . The lines and arrows “carry” variables ( $Y_i$  in our case) among the systems. In addition to system blocks, sometimes **summing junctions**, as shown on the right of [Fig. intro.3](#), sum two variables; in our example,  $Y_6 = Y_4 + Y_5$ .

## Time-domain (differential) equations

**11** Now we begin to make explicit all the mathematics implied in the graphical representations above. We have already seen how algebraic relationships are implied by the interconnection of elements in schematics, linear graphs, and block diagrams. The “elemental” equations implicit in schematics and linear graphs can also be algebraic, but usually some in every system are **differential equations**. For example, the elemental equation for a mass element  $m$  is Newton’s second law, which, in one dimension and with applied force  $f$  and coordinate  $x$ , is

$$f = m \frac{d^2x}{dt^2}.$$

The time-derivative here makes this a differential equation. When combined with algebraic and other differential equations in the system, the system of equations remains differential.

**12** So a system is described by a system of algebraic and differential equations. There are two common ways to represent this system. The first is as a single differential equation of order  $n$ , which is the result of combining all the algebraic and differential equations into a single scalar equation. The result—when the system is ordinary, linear, and time-invariant (all these terms will be described later in this text)—is what we call the **input-output differential equation** (io ODE) that relates input variable  $u(t)$  and output variable  $y(t)$ :

$$\begin{aligned} \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = \\ b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u \end{aligned} \quad (1)$$

where  $a_i, b_j$  are constants defined in terms of a system’s parameters. The io ODE representation is convenient because, for many common system orders  $n$  and inputs  $u$ , the analytic **solution** for  $y(t)$  is known, or at least a methodical process for deriving the solution is known. [Chapters 05 lti](#) and [06 trans](#) make extensive use of the io ODE representation and show multiple solution techniques.

**13** The other common representation of the system of equations is the **state equation**: a system of first-order ODEs in dependent variables  $x_i(t)$

collected into a state vector  $\mathbf{x}(t)$  and multiple input variables  $u_j(t)$  collected into the input vector  $\mathbf{u}$  such that

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (2a)$$

where  $\mathbf{f}$  is a vector-valued function that also depends on the system parameters. This equation can be linear or nonlinear. Note that multiple outputs  $y_k$  can also be found from the **output equation**

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \quad (2b)$$

where  $\mathbf{g}$  is a vector-valued function that also depends on the system parameters.

**14** Together, Eqs. 2a and 2b comprise what is called a **state-space model** of a system. The state-space model can fully represent a system with multiple inputs and multiple outputs, something not possible with an io ODE. For many linear systems, the state equation can be solved analytically, as described in ???. Another advantage of the state-space model is that it can also be easily solved **numerically**, as ??? also covers.

**15** The mathematical system representations above include many variables that are presented as explicit functions of time  $t$ , which is why we say that they are **time-domain** representations.

### Frequency-domain (algebraic) equations

**16** The remaining system representations are in what is called the **frequency domain**, which encompasses those that involve functions of not time but either angular frequency  $\omega$  or **Laplace transform** variable  $s$ . In fact, the Laplace and **Fourier transforms**<sup>2</sup> and their inverses are the bridges between the time- and frequency-domains.

**17** The frequency domain will be properly introduced in ???. For now, let's define these two remaining system representations in terms of their respective transforms. It is worth noting that the following representations are only defined for linear systems.

<sup>2</sup>The Fourier series is also included, here.

**18** The **transfer function**  $H(s)$  is defined as the ratio of the Laplace transform of the output  $Y(s) \equiv \mathcal{L}(y(t))$  to the Laplace transform of the input  $U(s) \equiv \mathcal{L}(u(t))$ ; i.e.

$$H(s) \equiv \frac{Y(s)}{U(s)}. \quad (3)$$

A quick rearrangement yields the output

$$Y(s) = H(s)U(s). \quad (4)$$

If we were to inverse Laplace transform  $\mathcal{L}^{-1}$  this, we would get  $y(t)$ ! Let's loop back for a moment to the block diagram representations of [Fig. intro.3](#). We said the block  $G$  maps input  $Y_1$  to output  $Y_2$ . The specific mapping is now clear: the block  $G$  is usually represented by the transfer function  $G(s)$ , so

$$G(s) = \frac{Y_2(s)}{Y_1(s)} \quad \text{and} \quad Y_2(s) = G(s)Y_1(s). \quad (5)$$

We see that this means, in the Laplace domain, system blocks are just algebraic products of the input and the transfer function. What's more, this transfer function thinking is very powerful: the system is understood as operating on an input with a transfer function and yielding an output.

**19** The **frequency response function**  $H(j\omega)$  is defined similarly, but in terms of the Fourier transform. The Fourier transform  $\mathcal{F}$  is introduced gently in [Chapter 09 four](#), but for now just think of it as similar to the Laplace transform (it is). The frequency response function  $H(j\omega)$  is defined as the ratio of the Fourier transform of the output  $Y(\omega) \equiv \mathcal{F}(y(t))$  to the Fourier transform of the input  $U(\omega) \equiv \mathcal{F}(u(t))$ ; i.e.

$$H(j\omega) \equiv \frac{Y(\omega)}{U(\omega)}. \quad (6)$$

A quick rearrangement yields the output

$$Y(\omega) = H(j\omega)U(\omega). \quad (7)$$

If we were to inverse Fourier transform  $\mathcal{F}^{-1}$  this, we would get  $y(t)$ ! Here we can gain the insight that the frequency response function, like the

transfer function, relates the input and output. The difference is that the frequency response function is explicitly dependent on the angular frequency  $\omega$ . We will see that, for a sinusoidal input at frequency  $\omega$ ,  $H(j\omega)$  returns a sinusoid at the same frequency, with only the amplitude and phase altered! This will be one of the key insights that will allow us to understand the performance of a system in terms of its frequency response function.

### Summarizing

**20** We've considered the three most important graphical and four most important explicitly mathematical dynamic system representations! Most of system dynamics is the study of such representations, their construction, characteristics, and performance. If you're overwhelmed, keep in mind that this is a high-level view of ... the entirety of system dynamics!

**21** Many of these representations can straightforwardly be converted into any other. The representations and their conversion pathways are sketched in [Fig. intro.4](#). Use this as a map as we explore the rich landscape of system dynamics!

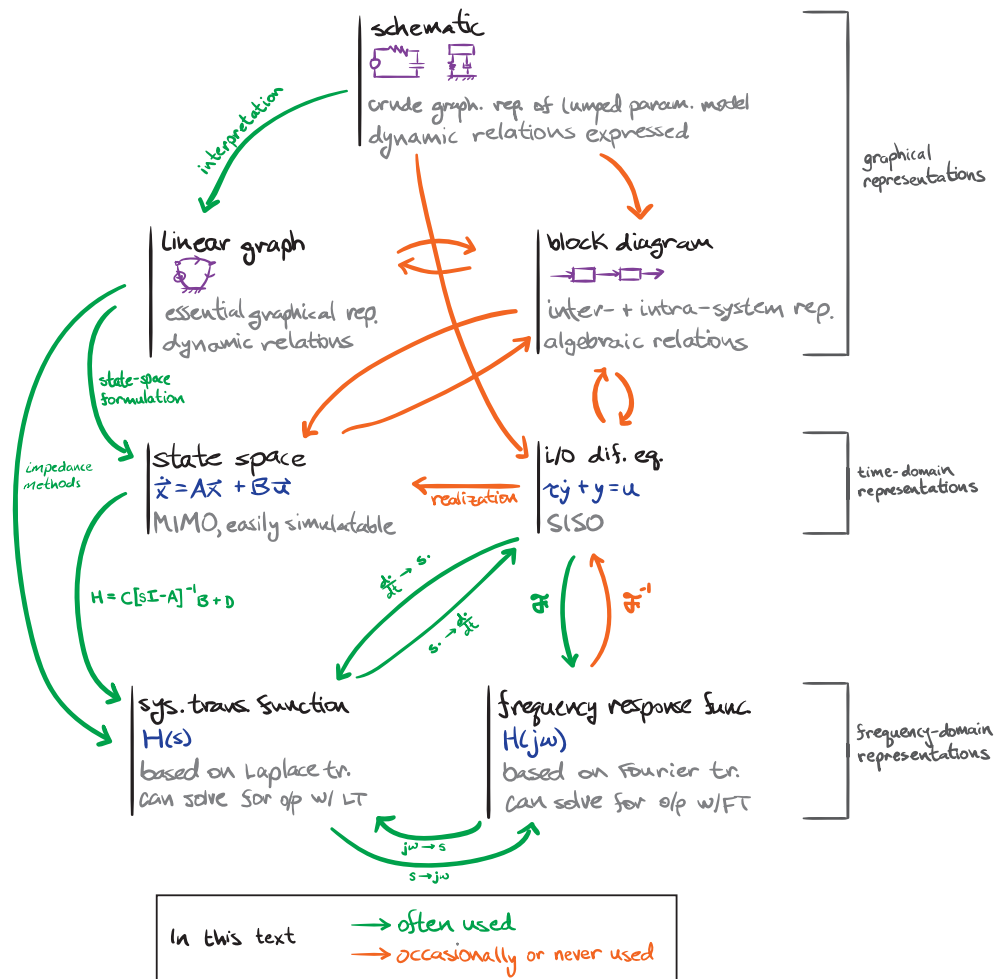


Figure intro.4: relations among system representations.

## 01.1 intro.it The systems approach

1 Simon Ramo and Richard Booton, Jr.—the folks who brought us the intercontinental ballistic missile (ICBM) (thanks? ...I mean thanks. But, thanks?)—defined **systems engineering** to be

*the design of the whole as distinguished from the design of the parts.*  
(*Boaton and Ramo, 1984*)

Like the ICBM, many modern technologies require this systems engineering design approach.

2 A key aspect of the systems engineering design process is the **mathematical modeling** of the system—the development of a dynamic system representation.

3 Dynamic systems exhibit behavior that can be characterized through analysis and called the system's **properties**. A property of a dynamic system might be how long it takes for it to respond to a given input or which types of inputs would cause a damaging response. Clearly, such properties are of significant interest to the design process.

4 This Part of the text focuses on **electromechanical systems**: systems with an interface between electronics and mechanical subsystems. These are ubiquitous: manufacturing plants, power plants, vehicles, robots, consumer products, anything with a motor—all include electromechanical systems. In ??, we will consider more types of systems (e.g. fluid and thermal) and their interactions.

5 Electromechanical systems analysis can proceed with initially separate modeling of the electronic and mechanical subsystems, then, through an unholy union, combining their equations and attempting a solution. This is fine for simple systems. However, many systems will require a systematic approach.

6 We adopt a systematic approach that draws **linear graphs** (à la **graph theory**) for electronic and mechanical systems that are intentionally analogous to electronic circuit diagrams. This allows us to use a single graphical diagram to express a system's composition and interconnections. Virtually every technique from electronic circuit analysis can be applied to



these representations. Elemental equations, Kirchhoff's laws, impedance—each will be generalized. In ?? , this same graphical and electronic-analog technique will be extended to other energy domains.

## 01.2 intro.sdet State-determined systems

**1** A **system** is defined to be a collection of objects and their relations contained within a **boundary**. The collection of those objects that are external to the system and yet interact with it is called the **environment**. **System variables** are variables that represent the behavior of the system, both those that are internal to the system and those that are external—that is, with the system’s environment.

**2** There are three important classes of system variable, all typically expressed as vector-valued functions of time  $t$ , conventionally all expressed as column-vectors (and called “vectors” even though they’re vector-valued functions ...because nothing makes sense and we’re all going to die). Consider [Figure sdet.1](#) for the following definitions. **Input variables** are system variables that do not depend on the internal dynamics of the system; for a system with  $r$  input variables, the “**input vector**” is a vector-valued function  $\mathbf{u} : \mathbb{R} \rightarrow \mathbb{R}^r$ . The environment prescribes inputs, making them *independent variables*. Conversely, **output variables** are system variables of interest to the designer; for a system with  $m$  output variables, the “**output vector**” is a vector-valued function  $\mathbf{y} : \mathbb{R} \rightarrow \mathbb{R}^m$ . Outputs may or may not directly interact with the environment. Finally, a minimal set of variables that define the internal state of the system are defined as the **state variables**; for a system with  $n$  state variables, the “**state vector**” is a vector-valued function  $\mathbf{x} : \mathbb{R} \rightarrow \mathbb{R}^n$ .

**3** We consider a very common class of system: those that are **state-determined**, which are those for which (Rowell and Wormley, 1997)

1. a mathematical description,
2. the state at time  $t_0$ , called the **initial condition**  $\mathbf{x}(t)|_{t=t_0}$ , and
3. the input  $\mathbf{u}$  for all time  $t \geq t_0$

are necessary and sufficient conditions to determine  $\mathbf{x}(t)$  (and therefore  $\mathbf{y}(t)$ ) for all  $t \geq t_0$ .

4 The “mathematical description” of the system requires a set of primitive elements be assigned to represent its internal and external interactions. The equations derive from two key types of mathematical relationships:

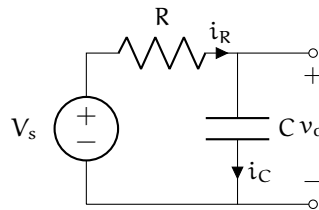
1. the input-output behavior of each primitive element and
2. the topology of interconnections among elements.

The former generate **elemental equations** and the latter, **continuity** or **compatibility equations**.

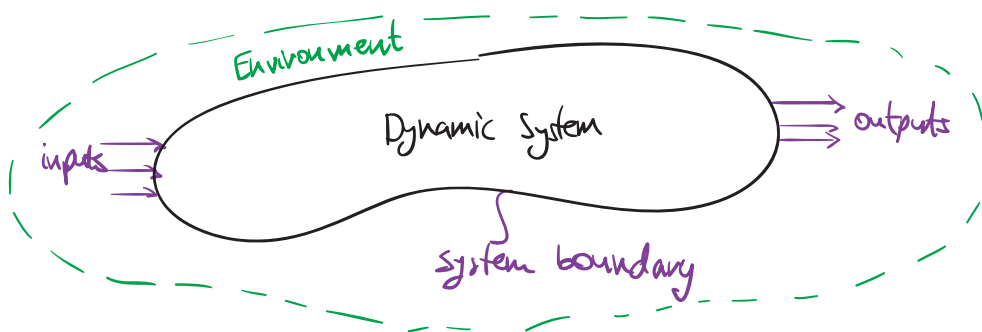
### Example 01.2 intro.sdet-1

In the RC circuit shown, let  $V_s$  be a source and  $v_o$  the voltage of interest. Identify

1. the system boundary,
2. the input vector,
3. the output vector,
4. a state vector,
5. an elemental equation,
6. and which equations might be continuity or compatibility equations.



re: a  
state-  
determined  
system



**Figure sdet.1:** illustration of a system and its environment.



## 01.3 intro.lump Energy, power, and lumping

**1** The **law of energy conservation** states that, for an isolated system, the total energy remains constant. Let  $\mathcal{E} : \mathbb{R} \rightarrow \mathbb{R}$  be the function of time representing the total energy in a system and  $\mathcal{P} : \mathbb{R} \rightarrow \mathbb{R}$  be the function of time representing **power** into the system, defined as

$$\mathcal{P}(t) = \frac{d\mathcal{E}(t)}{dt}. \quad (1)$$

The energy in a system can change if it exchanges energy with its environment. We consider this exchange to occur through a finite number of *ports*, each of which can supply or remove energy (positive or negative power), as in [Figure lump.1](#). This is expressed in an equation for power into a port  $\mathcal{P}_i$  and  $N$  ports as



We construct our systems such that they have no internal energy sources.

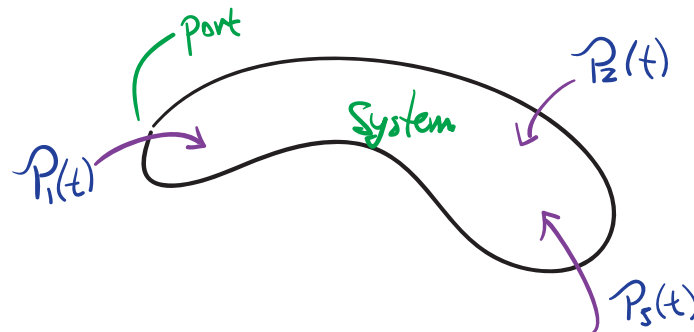
### Lumping

**2** We have assumed power enters a system via a finite number of ports. Similarly, we assume the energy *in* a system is stored in a finite number  $M$  of distinct elements with energy  $\mathcal{E}_i$  such that

$$\mathcal{E}(t) = \sum_{i=1}^M \mathcal{E}_i(t). \quad (2)$$

We call these elements **energy storage elements**. Energy can also be dissipated from the system via certain elements called **energy dissipative elements**.

**3** Considering a system to have a finite number of elements, as we have done, requires a specific kind of **abstraction** from real systems. A familiar



**Figure lump.1:** system ports

example is the “point mass” of elementary mechanics. We say it interacts with its environment via specific connections called ports (maybe it’s attached to a spring element) and behaves a certain way in these interactions (for a mass element, Newton’s second law). We do not often encounter an object that behaves as if it has mass, but no volume. Yet, this is a useful abstraction for many problems.

4 When we abstract like this, considering an object to be described fully as a discrete object with interaction ports, we are said to be **lumped-parameter modeling**. This is often contrasted with **distributed** or **continuous modeling**, which consider the element in greater detail. For instance, an object might be considered to be distributed through space and perhaps be flexible or behave as a fluid.

5 It is lumped-*parameter* modeling because we typically define a parameter that governs the behavior of the element, such as resistance or mass. This parameter will enter the system’s dynamics via an **elemental equation** such as Ohm’s Law in the case of a resistor or Newton’s Second Law in the case of a mass.

6 Determining if lumped-parameter modeling is proper for a given system is dependent on the type of insight one wants to achieve about the system. The system itself does not prescribe the proper modeling technique, but the desired understanding does. Every system is incredibly complex in its behavior, if one considers it at a fine-granularity. In this light, it is striking that simple models work *at all*. Nevertheless, lumping is highly effective for

many analyses.

7 It is important to note that lumped-parameter models can be applied at different levels of granularity for the same system. **Finite element modeling** can use a large number of small lumped-parameter model elements to approximate a continuous model. Such applications are beyond the scope of this course.

## 01.4 intro.mecht **Mechanical translational elements**

**1** We now introduce a few lumped-parameter elements for mechanical systems in translational (i.e. straight-line) motion. Newton's laws of motion can be applied. Let a **force**  $f$  and **velocity**  $v$  be input to a port in a mechanical translational element. Since, for mechanical systems, the power into the element is

$$\mathcal{P}(t) = f(t)v(t) \quad (1)$$

we call  $f$  and  $v$  the **power-flow variables** for mechanical translational systems. Some mechanical translational elements have two distinct locations between which its velocity is defined (e.g. the velocity across a spring's two ends) and other elements have just one (e.g. a point-mass), the velocity of which must have an inertial frame of reference. This is analogous to how a point in a circuit can be said to have a voltage—by which we mean “relative to ground.” In fact, we call this mechanical translational inertial reference **ground**.

**2** **Work** done on the system over the time interval  $[0, T]$  is defined as

$$W \equiv \int_0^T \mathcal{P}(\tau) d\tau. \quad (2)$$

Therefore, the work done on a mechanical system is

$$W = \int_0^T f(\tau)v(\tau) d\tau. \quad (3)$$

**3** The **linear displacement**  $x$  is

$$x(t) = \int_0^t v(\tau) d\tau + x(0). \quad (4)$$

Similarly, the **linear momentum** is

$$p(t) = \int_0^t f(\tau) d\tau + p(0). \quad (5)$$



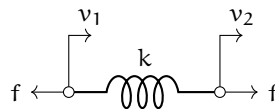
4 We now consider two elements that can store energy, called **energy storage elements**; an element that can dissipate energy to a system's environment, called an **energy dissipative element**; and two elements that can supply power from outside a system, called **source elements**.

### Translational springs

5 A **translational spring** is defined as an element for which the displacement  $x$  across it is a monotonic function of the force  $f$  through it. A **linear translational spring** is a spring for which Hooke's law applies; that is, for which

$$f(t) = kx(t), \quad (6)$$

where  $f$  is the force through the spring and  $x$  is the displacement across the spring, minus its unstretched length, and  $k$  is called the **spring constant** and is typically a function of the material properties and geometry of the element. This is called the element's **constitutive equation** because it constitutes what it means to be a spring.



**Figure mecht.1:** schematic symbol for a spring with spring constant  $k$  and velocity drop  $v = v_1 - v_2$ .

6 Although there are many examples of nonlinear springs, we can often use a linear model for analysis in some operating regime. The **elemental equation** for a linear spring can be found by time-differentiating Equation 6 to obtain



We call this the elemental equation because it relates the element's power-flow variables  $f$  and  $v$ .

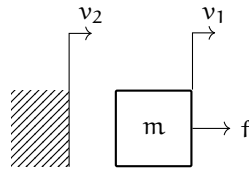
7 A spring stores energy as elastic potential energy, making it an *energy storage element*. The amount of energy it stores depends on the force it applies. For a linear spring,

$$\mathcal{E}(t) = \frac{1}{2k} f(t)^2. \quad (7)$$

### Point-masses

8 A non-relativistic translational point-mass element with mass  $m$ , velocity  $v$  (relative to an inertial reference frame), and momentum  $p$  has the constitutive equation

$$p = mv. \quad (8)$$



**Figure mecht.2:** schematic symbol for a point-mass with mass  $m$  and velocity drop  $v = v_1 - v_2$ , where  $v_2$  is the constant reference velocity.

Once again, time-differentiating the constitutive equation gives us the elemental equation:

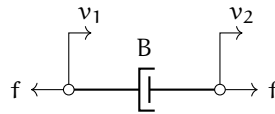


which is just Newton's second law.

9 Point-masses can store energy (making them *energy storage elements*) in gravitational potential energy or, as will be much more useful in our analyses, in kinetic energy

$$\mathcal{E}(t) = \frac{1}{2} mv^2. \quad (9)$$

## Dampers



**Figure mecht.3:** schematic symbol for a damper with damping coefficient  $B$  and velocity drop  $v = v_1 - v_2$ .

**10 Dampers** are defined as elements for which the force  $f$  through the element is a monotonic function of the velocity  $v$  across it. **Linear dampers** have constitutive equation (and, it turns out, elemental equation)

$$f = Bv \quad (10)$$

where  $B$  is called the **damping coefficient**. Linear damping is often called **viscous damping** because systems that push viscous fluid through small orifices or those that have lubricated sliding are well-approximated by this equation. For this reason, a damper is typically schematically depicted as a **dashpot**.

**11** Linear damping is a reasonable approximation of lubricated sliding, but it is rather poor for **dry friction** or **Coulomb friction**, forces for which are not very velocity-dependent. Aerodynamic **drag** is quite velocity-dependent, but is rather nonlinear, often represented as



where  $c$  is called the drag constant.

**12** Dampers dissipate energy from the system (typically to heat), making them *energy dissipative elements*.

## Force and velocity sources

**13** An **ideal force source** is an element that provides arbitrary energy to a system via an independent (of the system) force. The corresponding velocity across the element depends on the system.

**14** An **ideal velocity source** is an element that provides arbitrary energy to a system via an independent (of the system) velocity. The corresponding force through the element depends on the system.

## 01.5 intro.mechr Mechanical rotational elements

**1** We now introduce a few lumped-parameter elements for mechanical systems in rotational motion. Newton's laws of motion, in their angular analogs, can be applied. Let a **torque**  $T$  and **angular velocity**  $\Omega$  be input to a port in a mechanical rotational element. Since, for mechanical rotational systems, the power into the element is

$$\mathcal{P}(t) = T(t)\Omega(t) \quad (1)$$

we call  $T$  and  $\Omega$  the **power-flow variables** for mechanical rotational systems. Some mechanical rotational elements have two distinct locations between which its angular velocity is defined (e.g. the angular velocity across a spring's two ends) and other elements have just one (e.g. a rotational inertia), the velocity of which must have an inertial frame of reference. This is analogous to how a point in a circuit can be said to have a voltage—by which we mean “relative to ground.” In fact, we call this mechanical rotational inertial reference **ground**.

**2** **Work** done on the system over the time interval  $[0, t_f]$  is defined as

$$W \equiv \int_0^{t_f} \mathcal{P}(\tau) d\tau. \quad (2)$$

Therefore, the work done on a mechanical system is

$$W = \int_0^{t_f} T(\tau)\Omega(\tau) d\tau. \quad (3)$$

**3** The **angular displacement**  $\theta$  is

$$\theta(t) = \int_0^t \Omega(\tau) d\tau + \theta(0). \quad (4)$$

Similarly, the **angular momentum** is

$$h(t) = \int_0^t T(\tau) d\tau + h(0). \quad (5)$$

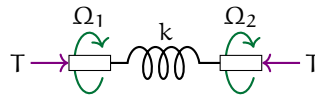
4 We now consider two elements that can store energy, called **energy storage elements**; an element that can dissipate energy to a system's environment, called an **energy dissipative element**; and two elements that can supply power from outside a system, called **source elements**.

### Rotational springs

5 A **rotational spring** is defined as an element for which the angular displacement  $\theta$  across it is a monotonic function of the torque  $T$  through it. A **linear rotational spring** is a rotational spring for which the angular form of Hooke's law applies; that is, for which

$$T(t) = k\theta(t), \quad (6)$$

where  $T$  is the torque through the spring and  $\theta$  is the angular displacement across the spring and  $k$  is called the **torsional spring constant** and is typically a function of the material properties and geometry of the element. This is called the element's **constitutive equation** because it constitutes what it means to be a rotational spring.



**Figure mechr.1:** schematic symbol for a spring with spring constant  $k$  and angular velocity drop  $\Omega = \Omega_1 - \Omega_2$ .

6 Although there are many examples of nonlinear springs, we can often use a linear model for analysis in some operating regime. The **elemental equation** for a linear spring can be found by time-differentiating Equation 6 to obtain



We call this the elemental equation because it relates the element's power-flow variables  $T$  and  $\Omega$ .

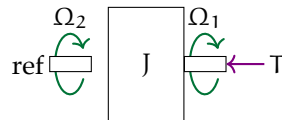
7 A rotational spring stores energy as elastic potential energy, making it an *energy storage element*. The amount of energy it stores depends on the torque it applies. For a linear rotational spring,

$$\mathcal{E}(t) = \frac{1}{2k} T(t)^2. \quad (7)$$

### Moments of inertia

8 A **moment of inertia** element with moment of inertia  $J$ , angular velocity  $\Omega$  (relative to an inertial reference frame), and angular momentum  $h$  has the constitutive equation

$$h = J\Omega. \quad (8)$$



**Figure mechr.2:** schematic symbol for a moment of inertia with inertia  $J$  and velocity drop  $\Omega = \Omega_1 - \Omega_2$ , where  $\Omega_2$  is a constant reference velocity.

Once again, time-differentiating the constitutive equation gives us the elemental equation:



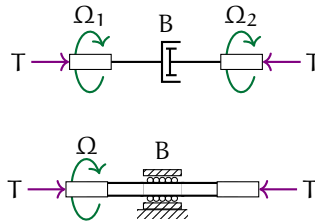
which is just the angular version of Newton's second law.

9 Any rotating element with mass can be considered as a lumped-inertia element. The **flywheel** is the quintessential example. Flywheels store energy in their angular momentum, with the expression

$$\mathcal{E}(t) = \frac{1}{2} J\Omega^2, \quad (9)$$

making them (and all moments of inertia) *energy storage elements*.

### Rotational dampers



**Figure mechr.3:** schematic symbol for a drag cup (above) and bearing (below) with damping coefficient  $B$ . For the drag cup, the angular velocity drop is  $\Omega = \Omega_1 - \Omega_2$  and for the bearing,  $\Omega$  is reference is ground.

**10 Rotational dampers** are defined as elements for which the torque  $T$  through the element is a monotonic function of the angular velocity  $\Omega$  across it. **Linear rotational dampers** have constitutive equation (and, it turns out, elemental equation)

$$T = B\Omega \quad (10)$$

where  $B$  is called the **torsional damping coefficient**. Linear torsional damping is often called **torsional viscous damping** because systems that push viscous fluid through small orifices or those that have lubricated bearings are well-approximated by this equation. For this reason, a damper is typically schematically depicted as a **drag cup** or as a **bearing**, both of which are shown in [Figure mechr.3](#).

**11** Linear damping is a reasonable approximation of lubricated sliding, but it is rather poor for **dry friction** or **Coulomb friction**, forces for which are not very velocity-dependent.

**12** Rotational dampers dissipate energy from the system (typically to heat), making them *energy dissipative elements*.

### Torque and angular velocity sources

**13** An **ideal torque source** is an element that provides arbitrary energy to a system via an independent (of the system) torque. The corresponding angular velocity across the element depends on the system.



**14** An **ideal angular velocity source** is an element that provides arbitrary energy to a system via an independent (of the system) angular velocity. The corresponding torque through the element depends on the system.

## 01.6 intro.ele Electronic elements

1 We now review a few lumped-parameter elements for electronic systems. Let a **current**  $i$  and **voltage**  $v$  be input to a port in an electronic element. Since, for electronic system, the power into the element is

$$P(t) = i(t)v(t) \quad (1)$$

we call  $i$  and  $v$  the **power-flow variables**. Voltage is always understood to be between two points in a circuit. If only one point is included, the voltage is implicitly relative to a reference voltage, called **ground**.

2 The **magnetic flux linkage**  $\lambda$  is

$$\lambda(t) = \int_0^t v(\tau) d\tau + \lambda(0). \quad (2)$$

Similarly, the **charge** is

$$q(t) = \int_0^t i(\tau) d\tau + q(0). \quad (3)$$

3 We now consider two elements that can store energy, called **energy storage elements**; an element that can dissipate energy to a system's environment, called an **energy dissipative element**; and two elements that can supply power from outside a system, called **source elements**.

### Capacitors

4 Capacitors have two terminal and are composed of two conductive surfaces separated by some distance. One surface has charge  $q$  and the other  $-q$ . A capacitor stores energy in an *electric field* between the surfaces.

5 Let a capacitor with voltage  $v$  across it and charge  $q$  be characterized by the parameter **capacitance**  $C$ , where the constitutive equation is

$$q = Cv. \quad (4)$$

6 The capacitance has derived SI unit **farad** ( $F$ ), where  $F = A \cdot s/V$ . A farad is actually quite a lot of capacitance. Most capacitors have capacitances best represented in  $\mu F$ , nF, and pF.

7 The time-derivative of this equation yields the  $v$ - $i$  relationship (what we call the “elemental equation”) for capacitors.

$$\frac{dv}{dt} = \frac{1}{C} i \quad (5)$$

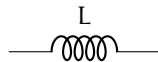
8 Capacitors allow us to build many new types of circuits: filtering, energy storage, resonant, blocking (blocks dc-component), and bypassing (draws ac-component to ground).

9 Capacitors come in a number of varieties, with those with the largest capacity (and least expensive) being **electrolytic** and most common being **ceramic**. There are two functional varieties of capacitors: **bipolar** and **polarized**, with circuit diagram symbols shown in Figure ele.1. Polarized capacitors can have voltage drop across in only one direction, from **anode** (+) to **cathode** (–)—otherwise they are damaged or may **explode**.

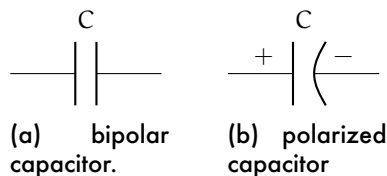
Electrolytic capacitors are polarized and ceramic capacitors are bipolar.

10 So what if you need a high-capacitance bipolar capacitor? Here’s a trick: place identical high-capacity polarized capacitors **cathode-to-cathode**. What results is effectively a bipolar capacitor with capacitance *half* that of one of the polarized capacitors.

## Inductors



**Figure ele.2:** inductor circuit diagram symbol.



**Figure ele.1:** capacitor circuit diagram symbols.

**11** A **pure inductor** is defined as an element in which **flux linkage**  $\lambda$ —the integral of the voltage—across the inductor is a monotonic function of the current  $i$ . An **ideal inductor** is such that this monotonic function is linear, with slope called the **inductance**  $L$ ; i.e. the ideal constitutive equation is

$$\lambda = Li \quad (6)$$

**12** The units of inductance are the SI derived unit **henry** ( $H$ ). Most inductors have inductance best represented in mH or  $\mu H$ .

**13** The elemental equation for an inductor is found by taking the time-derivative of the constitutive equation.

$$\frac{di}{dt} = \frac{1}{L}v \quad (7)$$

**14** Inductors store energy in a *magnetic field*. It is important to notice how inductors are, in a sense, the *opposite* of capacitors. A capacitor's current is proportional to the time rate of change of its voltage. An inductor's voltage is proportional to the time rate of change of its current.

**15** Inductors are usually made of wire coiled into a number of turns. The geometry of the coil determines its inductance  $L$ .

**16** Often, a **core** material—such as iron and ferrite—is included by wrapping the wire around the core. This increases the inductance  $L$ .

**17** Inductors are used extensively in radio-frequency (rf) circuits, which we won't discuss in this text. However, they play important roles in ac-dc conversion, filtering, and transformers—all of which we will consider extensively.

**18** The circuit diagram for an inductor is shown in [Figure ele.2](#).

## Resistors



**Figure ele.3:** resistor circuit diagram symbol.

**19** Resistors *dissipate* energy from the system, converting electrical energy to thermal energy (heat). The **constitutive equation** for an ideal resistor is

$$v = iR. \quad (8)$$

This is already in terms of power variables, so it is also the **elemental equation**.

### Sources

**20** Sources (a.k.a. supplies) supply power to a circuit. There are two primary types: *voltage sources* and *current sources*.

#### *Ideal voltage sources*

**21** An ideal voltage source provides exactly the voltage a user specifies, independent of the circuit to which it is connected. All it must do in order to achieve this is to supply whatever current necessary.

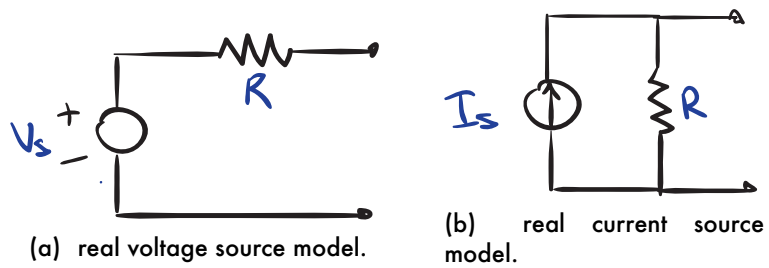
#### *Ideal current sources*

**22** An ideal current source provides exactly the current a user specifies, independent of the circuit to which it is connected. All it must do in order to achieve this is to supply whatever voltage necessary.

#### *Modeling real sources*

**23** No real source can produce infinite power. Some have feedback that controls the output within some finite power range. These types of sources can be approximated as ideal when operating within its specifications. Many voltage sources (e.g. batteries) do not have internal feedback controlling the voltage. When these sources are “loaded” (delivering power) they cannot maintain their nominal output, be that voltage or current. We model these types of sources as ideal sources in series or parallel with a resistor, as illustrated in [Figure ele.4](#).

**24** Most manufacturers specify the nominal resistance of a source as the “output resistance.” A typical value is 50  $\Omega$ .



**Figure ele.4:** Models for power-limited "real" sources.

## 01.7 intro.genvars **Generalized through- and across-variables**

1 We have considered mechanical translational, mechanical rotational, and electronic systems—which we refer to as different **energy domains**. There are analogies among these systems that allow for generalizations of certain aspects of these systems. These generalizations will allow us to use a single framework for unifying the analysis of these (and other) dynamic systems.

2 There are two important classes of variables common to lumped-parameter dynamic systems: *across-variables* and *through-variables*.

3 An **across-variable** is one that makes reference to two nodes of a system element. For instance, the following are across-variables:

- 
- 
- 

We denote a **generalized across-variable** as  $\mathcal{V}$ .

4 A **through-variable** is one that represents a quantity that passes through a system element. For instance, the following are through-variables:

- 
- 
- 

We denote a **generalized through-variable** as  $\mathcal{F}$ .

5 The **generalized integrated across-variable**  $\mathcal{X}$  is

$$\mathcal{X} = \int_0^t \mathcal{V}(\tau) d\tau + \mathcal{X}(0). \quad (1)$$

6 The **generalized integrated through-variable**  $\mathcal{H}$  is

$$\mathcal{H} = \int_0^t \mathcal{F}(\tau) d\tau + \mathcal{H}(0). \quad (2)$$

7 For mechanical and electronic systems, power  $\mathcal{P}$  passing through a lumped-parameter element is

$$\mathcal{P}(t) = \mathcal{F}(t)\mathcal{V}(t). \quad (3)$$

8 These generalized across- and through-variables are sometimes used in analysis. However, the key idea here is that there are two classes of power-flow variables: across and through. These two classes allow us to strengthen the sense in which we consider different dynamic systems to be analogous.



## 01.8 intro.genels Generalized one-port elements

1 We can categorize the behavior of one-port elements—electronic, mechanical translational, and mechanical rotational—considered thus far. In the following sections, we consider two types of energy storage elements, dissipative elements, and source elements.

### A-type energy storage elements

2 An element that stores energy as a function of its across-variable is called an **A-type energy storage element**. Sometimes we call it a **generalized capacitor** because a capacitor is an A-type energy storage element.

3 For generalized through-variable  $\mathcal{F}$ , across-variable  $\mathcal{V}$ , integrated through-variable  $\mathcal{H}$ , and integrated across-variable  $X$  the ideal, linear constitutive equation is

$$\mathcal{H} = C\mathcal{V} \quad (1)$$

for  $C \in \mathbb{R}$  called the **generalized capacitance**. Differentiating [Equation 1](#) with respect to time, the elemental equation is



A-type energy storage elements considered thus far are **capacitors**, **translational masses**, and **rotational moments of inertia**. As with generalized variables, the analogs among elements are more important than are generalized A-type energy storage elements.

### T-type energy storage elements

4 An element that stores energy as a function of its through-variable is called a **T-type energy storage element**. Sometimes we call it a **generalized inductor** because an inductor is a T-type energy storage element.

5 The ideal, linear constitutive equation is

$$\mathcal{X} = L\mathcal{F} \quad (2)$$

for  $L \in \mathbb{R}$  called the **generalized inductance**. Differentiating Equation 2 with respect to time, the elemental equation is



6 T-type energy storage elements considered thus far are **inductors**, **translational springs**, and **rotational springs**. As with generalized variables, the analogs among elements are more important than are generalized T-type energy storage elements.

### D-type energy dissipative elements

7 An element that dissipates energy from the system and has an algebraic relationship between its through-variable and its across-variable is called a **D-type energy dissipative element**. Sometimes we call it a **generalized resistor** because a resistor is a D-type energy dissipative element.

8 The ideal, linear constitutive and elemental equation is

$$\mathcal{V} = R\mathcal{F} \quad (3)$$

for  $R \in \mathbb{R}$  called the **generalized resistance**.

9 D-type energy dissipative elements considered thus far are **resistors**, **translational dampers**, and **rotational dampers**. As with generalized variables, the analogs among elements are more important than are generalized D-type energy dissipative elements.

### Sources

10 An **ideal through-variable source** is an element that provides arbitrary energy to a system via an independent (of the system) through-variable. The corresponding across-variable depends on the system. Current, force, and torque sources are the through-variable sources considered thus far.

**11** An **ideal across-variable source** is an element that provides arbitrary energy to a system via an independent (of the system) across-variable. The corresponding through-variable depends on the system. Voltage, translational velocity, and angular velocity are the across-variable sources considered thus far.

## 01.9 intro.exe Exercises for Chapter 01 intro

### Exercise 01.1 madrid

Consider the drivetrain of a standard internal combustion engine vehicle. When accelerating from a stop in wet weather it is common for the wheels to slip due to a film of water between the wheels and the road. Develop a lumped parameter model of this system with the following assumptions,

- the engine and transmission together can be simulated as a torque source,
- the transmission and wheels are connected with a drive shaft of finite stiffness, and
- each wheel has equal mass.

From this description please,

1. draw a one dimensional lumped parameter model (like the diagrams in problem granda), and
2. draw a linear graph of the lumped parameter model.

\_\_\_\_\_/

15 p.

## 02 graphs

---

## 02.1 graphs.intro Introduction to linear graphs

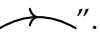
- 1 Engineers often use graphical techniques to aid in analysis and design. We will use **linear graphs** to represent the **topology** or structure of a system modeled as interconnected lumped elements.
- 2 This represents to us the essential structure of the system in a minimalist form. In this way, it is like Massimo Vignelli's famous 1972 New York

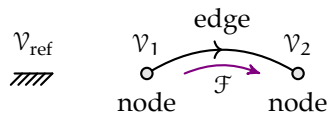


**Figure intro.1:** a modern New York subway map in the style of Vignelli (Jake Berman).

subway system “map,” which inspired widespread adoption of his style (see [Figure intro.1](#)).<sup>1</sup> Besides minimalism, the key idea in Vignelli subway maps is that the details of the tunnels’ paths are irrelevant and, in fact, distracting to the person attempting to get from one station to another.

**3** In a similar way, a linear graph represents the system in a minimalist style, with only two types of objects:

1. A set of **edges**, each of which represents an energy port associated with a system element. Each edge is drawn as an oriented line segment “”.
2. A set of **nodes**, each of which represents a point of interconnection among system elements. Each node is drawn as a dot “ $\circ$ ”.



**Figure intro.2:** an edge with nodes. The across variable is  $v = v_1 - v_2$ .

**4** All edges begin and end at nodes. The nodes represent locations in the system where distinct across-variable values may be measured. For example, wires that connect elements are actually *nodes* at which voltage may be measured. Putting an edge together with nodes, we have [Figure intro.2](#).

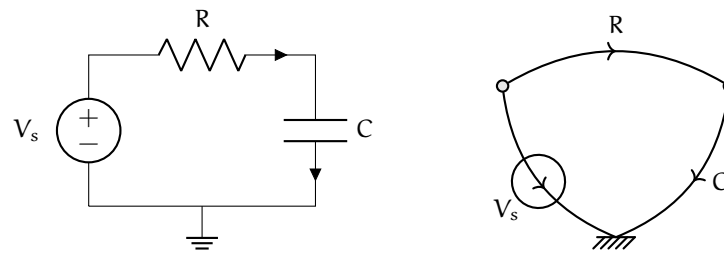
**5** It is important to note that linear graphs can represent **nonlinear** system elements—the name is a reference to the *lines* used.

**6** It is common to choose a node of the graph as the **reference node**, to which all across-variables are referenced. Due to its similarity to the electronic **ground**, we often use these terms interchangeably.

**7** [Figure intro.3](#) shows how a linear graph can be constructed for a simple RC-circuit. Note that the wires become nodes, the elements become edges, and the reference node represents the circuit ground. In a similar manner,

<sup>1</sup>Vignelli was a brilliant Minimalist designer of many products, from dishes to clothing, but he was most known for his graphic design. Great places to start studying Vignelli are the documentary *Design is One* (2012) and *The Vignelli Canon*.

we will construct linear graphs of circuits, mechanical translational systems, and mechanical rotational systems.



**Figure intro.3:** an example of a linear graph representation of an RC-circuit.



## 02.2 graphs.sign Sign convention

- 1 The **sign** (positive or negative) of a variable is used to represent an orientation of its physical quantity. For instance,  $-3$  m/s could mean 3 m/s to the *right* or *left*. No one can say which is better (right is better). Deciding how the physical quantity corresponds to the sign of the variable is called **sign assignment**. When we use a **sign convention**, we make the assignment in a conventional manner. For instance, the sign convention for normal stress is that compression is negative and tension is positive.
- 2 Why use a sign convention? If we follow a convention when constructing a problem, we can use the convention's **interpretation** of the result. For complicated systems, this helps us keep things straight. Furthermore, if someone else attempts to understand our work, it is much easier to simply say "using the standard sign convention, ..." than explaining our own snowflake sign assignment. However, it is nonetheless true that we can assign signs arbitrarily.
- 3 Arbitrary? **Vive la révolution!** But wait. If a *source* is present, we must observe some caution. A source typically *comes with its own convention*. For instance, if we hook up a power supply to the circuit with the + and – leads a certain way, unless we want to get very confused, we should probably accommodate that sign.
- 4 A sign convention for each of the energy domains we've considered follows.

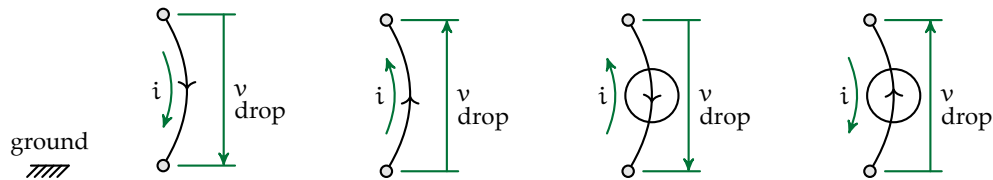
### Electronic systems

- 5 We use the **passive sign convention** of electrical engineering, defined below.

#### Definition 02 graphs.1: passive sign convention

Power flowing *in* to a component is considered to be *positive* and power flowing *out* of a component is considered *negative*.

- 6 Because power  $\mathcal{P} = vi$ , this implies the current and voltage signs are prescribed by the convention. For **passive elements**, the electrical potential



**Figure sign.1:** passive sign convention for electronic systems in terms of voltage  $v$  and current  $i$ . Passive elements are on the left, active on the right.

must drop in the direction of positive current flow. This means the assumed direction of voltage drop across a passive element must be the same as that of the current flow. For **active elements**, which supply power to the circuit, the converse is true: the voltage drop and current flow must be in opposite directions. [Figure sign.1](#) illustrates the possible configurations.

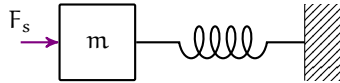
**7** When drawing a linear graph of a circuit, for each passive element's edge, draw the arrow beside it pointing in the direction of assumed current flow and voltage drop.

**8** The purpose of a sign convention is to help us **interpret** the signs of our results. For instance, if, at a given instant, a capacitor has voltage  $v_C = 3\text{ V}$  and current  $i_C = -2\text{ A}$ , we compute  $\mathcal{P}_C = -6\text{ W}$  and we know  $6\text{ W}$  of power is flowing *from* the capacitor into the circuit.

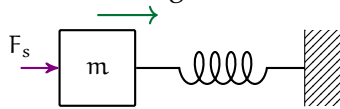
**9** For passive elements, there is no preferred direction of "assumed" voltage drop and current flow. If a voltage or current value discovered by performing a circuit analysis is positive, this means the "assumed" and "actual" directions are the same. For a negative value, the directions are opposite.

**10** For active elements, choose the sign in accordance with the physical situation. For instance, if a positive terminal of a battery is connected to a certain terminal in a circuit, it ill behooves one to simply say "but Darling, I'm going to call that negative." It's positive whether you like it or not, Nancy.

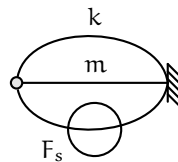
### Translational mechanical systems



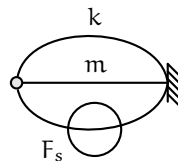
**11** The following steps can be applied to any translational mechanical system. We introduce the convention with an inline example. Consider the simple mechanical system shown at right.



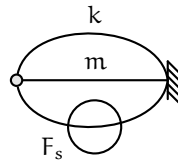
**12 coordinate arrow** Assign the sign by drawing a coordinate arrow, as shown at right. The direction of the arrow is arbitrary, however, if possible, assign the positive direction to match the sources. If the problem allows, it is best practice to have all sources and the coordinate arrow in the same direction.



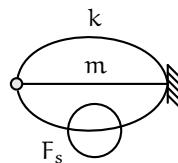
**draw linear graph without arrows** There are two nodes with distinct velocities: ground and the mass, as shown at right. The mass node is always drawn to ground. The spring connects between the mass and ground. Finally, the force source connects to the mass, where it is applied, and also connects to ground, which is impervious to it.



**13 assign spring and damper directions** On each spring and damper element, define the positive velocity drop and edge arrow to be *in the direction of the coordinate arrow*.



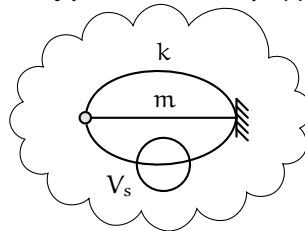
**14 assign mass directions** On each mass element, define the positive velocity drop and edge arrow to be *toward ground*. Sometimes we dash the latter half of the mass edge in to signify that it is “virtually” connected to ground.



**15 assign force source directions** On each force source element, define the positive direction as follows.

(ideal) If the force source has the *same* definition of positive as your coordinate arrows, draw it *toward the node of application*.

(if needed) If the force source has the *opposite* definition of positive as your coordinate arrow, draw it *away from the node of application*.



**16 assign velocity source directions** On each velocity source element, define the positive direction as follows.

(ideal) If the velocity source has the *same* definition of positive as your coordinate arrows, draw it *away from the node of application*.

(if needed) If the velocity source has the *opposite* definition of positive as your coordinate arrow, draw it *toward the node of application*.

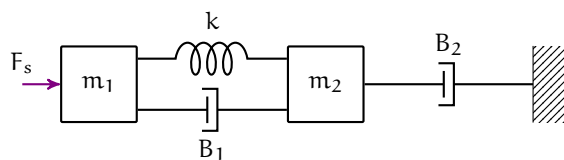
**17** This convention yields the interpretations of [Table sign.1](#).

**Table sign.1:** interpretation of the translational mechanical system sign convention.

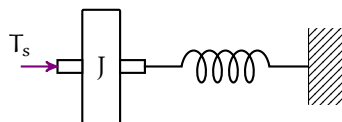
		force $f$		velocity $v$	
		positive +	negative -	positive +	negative -
$m$		force <i>in</i> direction of the coordinate arrow	force <i>opposite</i> the direction of the coordinate arrow	velocity <i>in</i> the coordinate arrow direction	velocity <i>opposite</i> the coordinate arrow direction
$k$		<i>compressive</i> force	<i>tensile</i> force	velocity drops <i>in</i> the coordinate arrow direction	velocity drops <i>opposite</i> the coordinate arrow direction
$B$		<i>compressive</i> force	<i>tensile</i> force	velocity drops <i>in</i> the coordinate arrow direction	velocity drops <i>opposite</i> the coordinate arrow direction

**Example 02.2 graphs.sign-1**

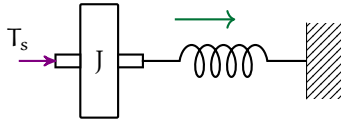
For the system shown, draw a linear graph and assign signs according to the sign convention.



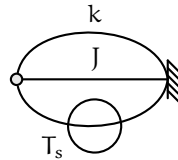
re:  
**translational  
mechanical  
sign  
convention**

**Rotational mechanical systems**

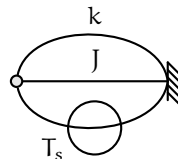
**18** The following steps can be applied to any rotational mechanical system. We introduce the convention with an inline example. Consider the simple system shown at right.



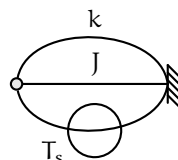
**19 coordinate arrow** Assign the sign by drawing a coordinate arrow, as shown at right. The direction of the arrow is arbitrary, however, if possible, assign the positive direction to match the sources. If the problem allows, it is best practice to have all sources and the coordinate arrow in the same direction. The right-hand rule is always implied.



**20 draw linear graph without arrows** There are two nodes with distinct velocities: ground and the inertia, as shown at right. The inertia node is always drawn to ground. The spring connects between the inertia and ground. Finally, the torque source connects to the mass, where it is applied, and also connects to ground, which is impervious to it.

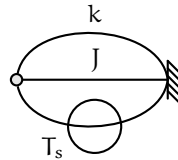


**21 assign spring and damper directions** On each inline spring and damper element, define the positive velocity drop and edge arrow to be *in the direction of the coordinate arrow*. Springs and dampers that aren't inline typically connect to ground, toward which edge arrows should point.



**22 assign inertia directions** On each inertia element, define the

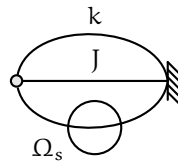
positive angular velocity drop and edge arrow to be *toward ground*. Sometimes we dash the latter half of the inertia edge to signify that it is “virtually” connected to ground.



**23 assign torque source directions** On each torque source element, define the positive direction as follows.

(ideal) If the torque source has the *same* definition of positive as your coordinate arrows, draw it *toward the node of application*.

(if needed) If the torque source has the *opposite* definition of positive as your coordinate arrow, draw it *away from the node of application*.



**24 assign angular velocity source directions** On each angular velocity source element, define the positive direction as follows.

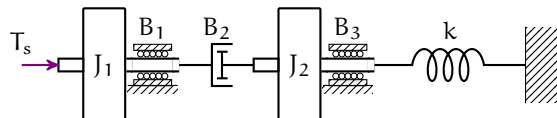
(ideal) If the source has the *same* definition of positive as your coordinate arrows, draw it *away from the node of application*.

(if needed) If the source has the *opposite* definition of positive as your coordinate arrow, draw it *toward the node of application*.

**25** This convention yields the interpretations of [Table sign.2](#).

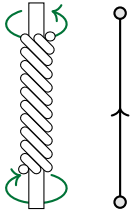
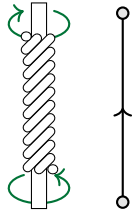
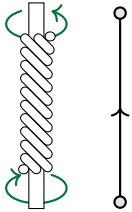
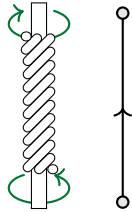
### Example 02.2 graphs.sign-2

For the system shown, draw a linear graph and assign signs according to the sign convention.



re:  
**rotational  
mechanical  
sign  
convention**


**Table sign.2:** interpretation of the mechanical system sign convention.

		torque $T$		angular velocity $\Omega$	
		positive +	negative -	positive +	negative -
J		torque <i>in</i> direction of the coordinate arrow	torque <i>opposite</i> the direction of the coordinate arrow	velocity <i>in</i> the coordinate arrow direction	velocity <i>opposite</i> the coordinate arrow direction
k	wring!		wrong! 	velocity drops <i>in</i> the coordinate arrow direction	velocity drops <i>opposite</i> the coordinate arrow direction
B	wring!		wrong! 	velocity drops <i>in</i> the coordinate arrow direction	velocity drops <i>opposite</i> the coordinate arrow direction





## 02.3 graphs.connect Element interconnection laws

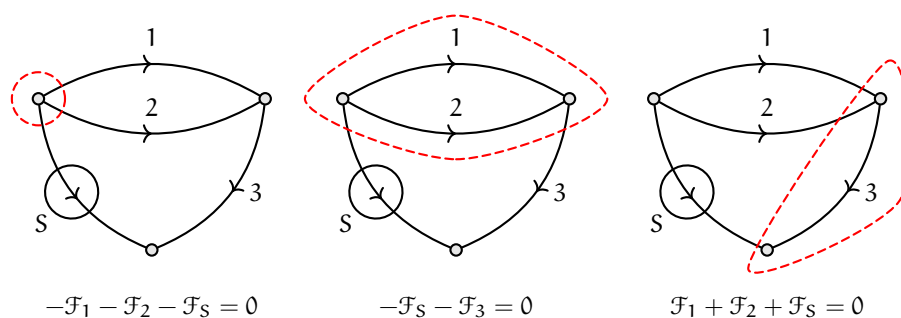
1 The interconnections among elements constrain across- and through-variable relationships. The first element interconnection law requires the concept of a **contour** “”: a closed path that does not self-intersect superimposed over the linear graph. The first interconnection law is called the **continuity law**.


### Definition 02 graphs.2: continuity law

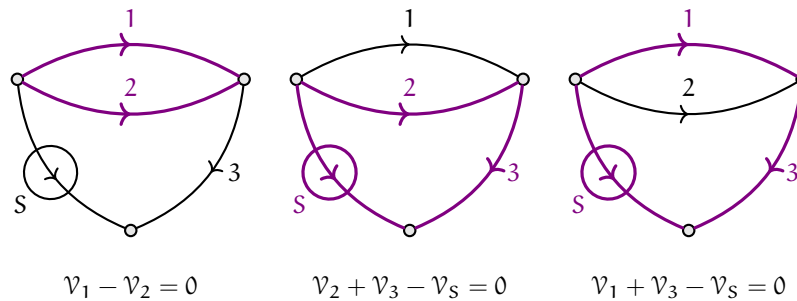
The sum of the through-variables that flow on *into* a contour on a linear graph is zero, or, in terms of generalized through-variables  $\mathcal{F}_i$  for  $N$  elements with through variables defined as positive into the contour,

$$\sum_{i=1}^N \mathcal{F}_i = 0. \quad (1)$$

2 Contours can enclose any number of nodes and edges, as illustrated in [Figure connect.1](#). **Kirchhoff's current law** (KCL) is the special case of the continuity law for electronic systems.



**Figure connect.1:** illustration of different contours, denoted with red dashed lines “,” contours for which the continuity law applies, as shown below each graph.



**Figure connect.2:** illustration of different loops, denoted with violet edges “”, loops for which the compatibility law applies.

3 The second interconnection law we consider requires the concept of a **loop** “”: a continuous series of edges that begin and end at the same node, not reusing any edges.<sup>2</sup> The second interconnection law is called the **compatibility law**.

#### Definition 02 graphs.3: compatibility law

The sum of the across-variable drops on edges around any closed loop on a linear graph is zero, or, in terms of generalized across variables  $v_i$  for  $N$  elements in a loop,

$$\sum_{i=1}^N v_i = 0. \quad (2)$$

A loop can be “inner” or “outer,” as shown in [Figure connect.2](#). **Kirchhoff’s voltage law** (KVL) is the special case of the compatibility law for electronic systems.

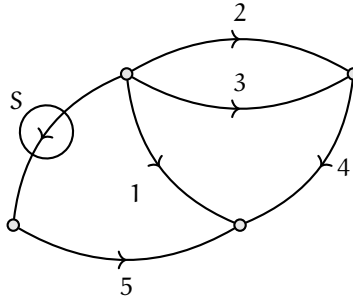
#### Example 02.3 graphs.connect-1

For the system shown, (a) write three unique continuity and three unique compatibility equations. Moreover, (b) write a continuity equation solved for  $\mathcal{F}_4$  in terms of  $\mathcal{F}_S$  and  $\mathcal{F}_1$ . Finally, (c) write a compatibility equation

re:  
element  
interconnection  
laws

<sup>2</sup>Technically, we need not restrict the definition to series that do not reuse edges for purposes of the compatibility law, but these loops are superfluous and we exclude them here.

• solved for  $v_5$  in terms of  $v_s$ ,  $v_3$ , and  $v_4$ .



## 02.4 graphs.sysmod Systematic linear graph modeling

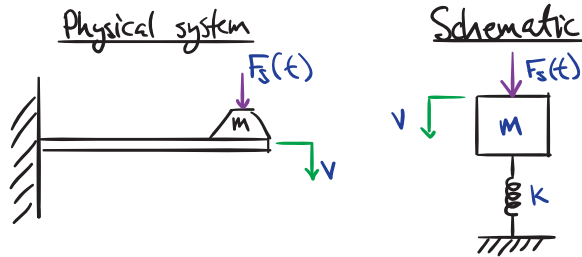
**1** A **system graph** is a representation of a physical system as a set of interconnected linear graph elements. The construction of a system graph requires a number of engineering decisions. In general, we can use the following procedure.

1. Define the system boundary and analyze the physical system to determine the essential features that must be included in the model, especially:
  - a) inputs,
  - b) outputs,
  - c) energy domains, and
  - d) key elements.
2. Form a schematic model of the physical system and assign schematic signs according to the sign convention of [Lecture 02.2 graphs.sign](#).
3. Determine the necessary lumped-parameter elements representing the system's
  - a) energy sources,
  - b) energy storage, and
  - c) energy dissipation.
4. Identify the across-variables that define the linear graph nodes and draw a set of nodes.
5. Determine appropriate nodes for each lumped element and include each element in the graph.
6. Assign linear graph signs according to the sign convention of [Lecture 02.2 graphs.sign](#).

**2** The first three of these steps are the hardest. Considerable physical insight is required to construct an effective model. Often it is helpful—if not necessary—to have experimental results to guide the process.

**Example 02.4 graphs.sysmod-1**

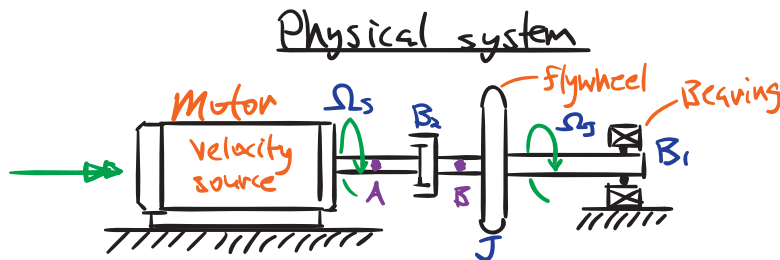
For the system shown, develop a linear graph model.



re:  
linear  
graph  
model  
of  
translational  
mechanical  
system

**Example 02.4 graphs.sysmod-2**

For the system shown, develop a linear graph model.

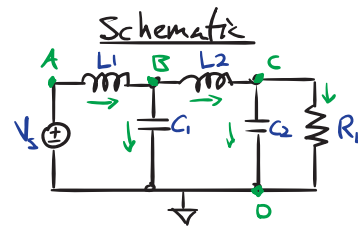
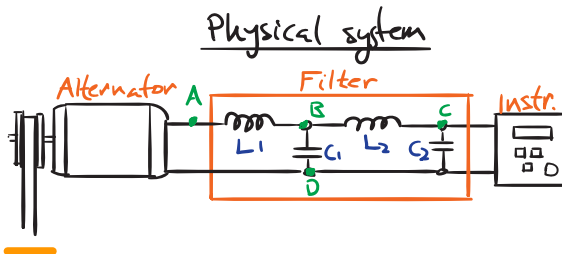


re:  
linear  
graph  
model  
of  
rotational  
mechanical  
system



**Example 02.4 graphs.sysmod-3**

For the system shown, develop a linear graph model.



re:  
linear  
graph  
model  
of  
electronic  
system

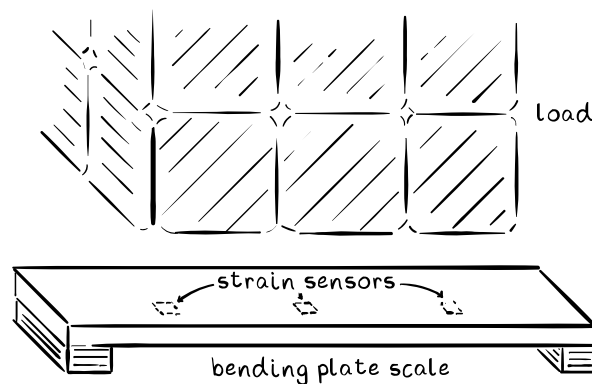
## 02.5 graphs.exe Exercises for Chapter 02 graphs

### Exercise 02.1 playmate

Consider the illustration of Fig. exe.1 in which a bending plate scale is to have a heavy load placed upon it. Such scales measure the weight of the load by measuring the strain on the sensors and electronically converting this to the weight placed on the plate. (It goes without saying that calibration is required for such systems.)

It takes time for the system to come to equilibrium, during which oscillation occurs. Develop a one-dimensional lumped-parameter model of the mechanical aspect of the system and its applied load, via the following steps.

1. Declare what you will take to be the system and its input(s).
2. Declare a one-dimensional, mechanical, lumped-parameter model for the system. How might you determine the lumped-parameter model parameters (e.g. mass, spring constant, etc.)?
3. Draw a schematic of the lumped-parameter system model.
4. Draw a linear graph corresponding to your lumped-parameter model.



**Figure exe.1:** a bending plate scale with strain sensors and load.

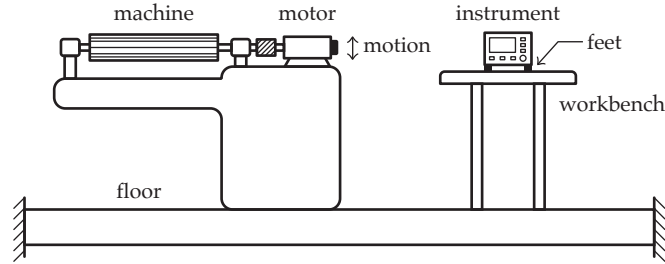
**Exercise 02.2 nod**

Consider the illustration of Fig. exe.2 in which a motor is on a machine and an instrument is atop a nearby workbench. The motor typically spins at a fixed velocity, generating a vibration that is transmitted through the machine and into the floor.

Suppose you are given the task of designing the feet of the instrument such that less than a certain amount of vibratory motion from the motor will be transmitted through the floor and workbench to the instrument.

Develop a one-dimensional lumped-parameter model of the mechanical aspect of the system via the following steps.

1. Declare what you will take to be the system and its input(s).
2. Declare a one-dimensional, mechanical, translational, lumped-parameter model for the system.
3. Draw a schematic of the lumped-parameter system model.
4. Draw a linear graph corresponding to your lumped-parameter model.



**Figure exe.2:** A motor on a machine and a nearby instrument on a workbench.

**Exercise 02.3 johnnycash**

Consider the illustration of Fig. exe.3 in which a wind turbine is harvesting wind energy. An electrical generator converts the rotational mechanical power into electrical energy.

Suppose you are given the task of designing the bearing and the flexible shaft coupler assemblies. For the design, you will need to know how wind speeds will affect the angular velocities and torques throughout the



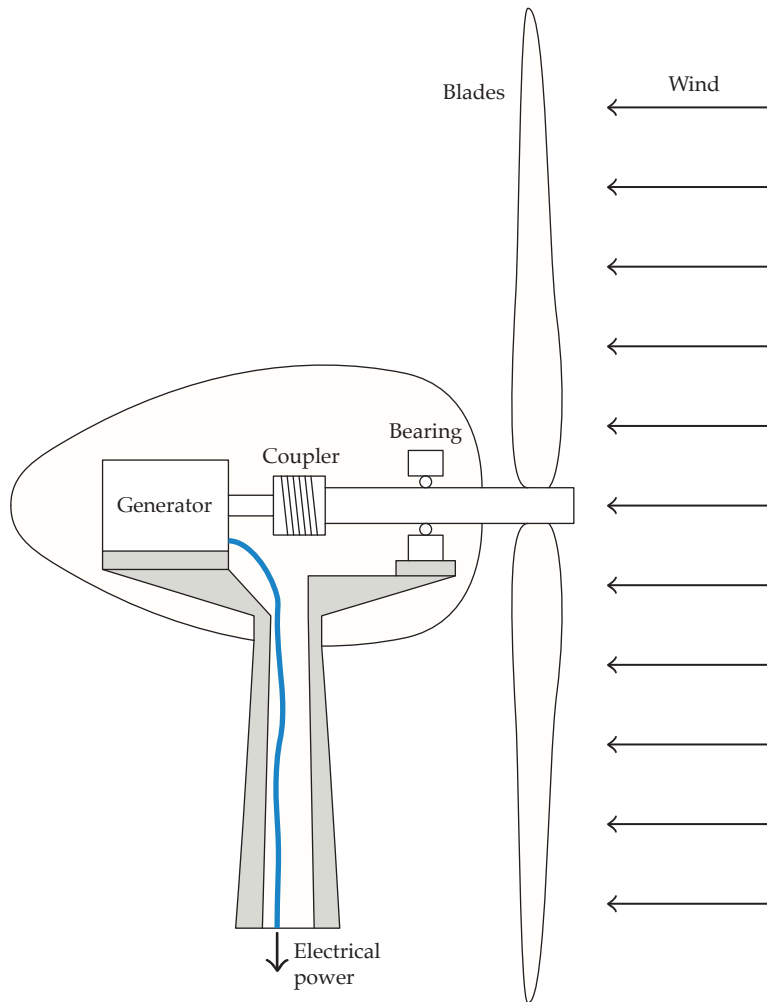
rotational mechanical system. Therefore, you resolve to develop a dynamic system model.

Develop a one-dimensional lumped-parameter model of the mechanical aspect of the system. Assume that the wind produces an input torque  $T_S$  via the turbine blades. Further assume that the generator draws power from the mechanical system in a manner that produces a load torque  $T_G$  that is proportional to the generator shaft angular velocity  $\Omega_G$ ; that is, for constant  $\beta$ ,

$$T_G = \beta\Omega_G. \quad (1)$$

Use the following steps:

1. Declare what you will take to be the system and its input(s).
2. Declare a one-dimensional, rotational mechanical, lumped-parameter model for the system.
3. Draw a schematic of the lumped-parameter system model.
4. Draw a linear graph corresponding to your lumped-parameter model.

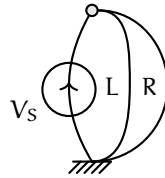


**Figure exe.3:** A sketch of a wind turbine.

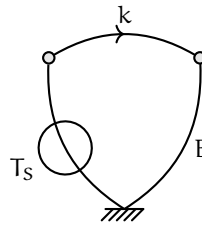
### Exercise 02.4 lillimoomie

Finish applying the sign coordinate arrows on the following linear graphs.

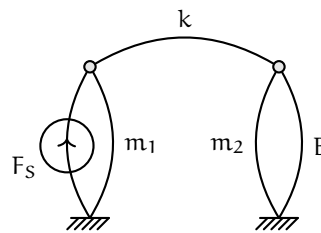
- electronic system



b. rotational mechanical system (assume  $T_s$  is in the positive direction)



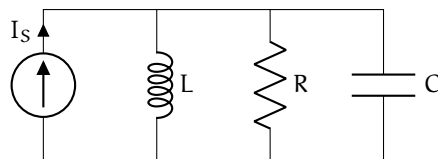
c. translational mechanical system



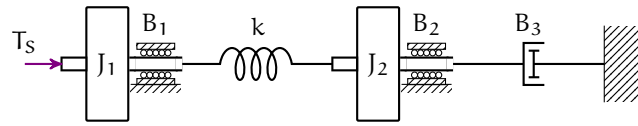
**Exercise 02.5 varieties**

Draw necessary sign coordinate arrows and a linear graph for each of the following schematics.

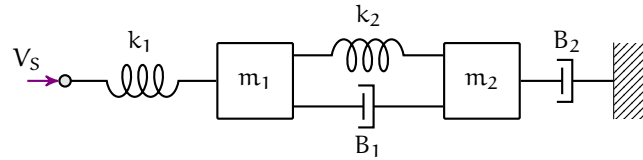
a. electronic system, current source



b. rotational mechanical system, torque source



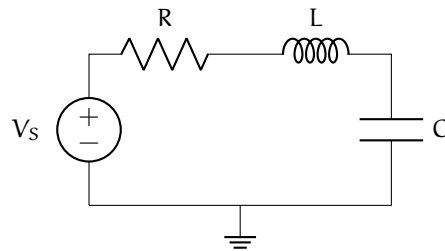
c. translational mechanical system, velocity source



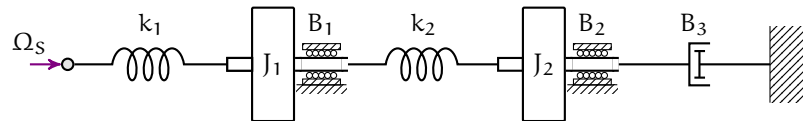
**Exercise 02.6 cormac**

Draw necessary sign coordinate arrows and draw a linear graph for each of the following schematics.

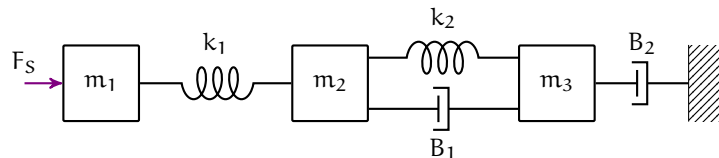
a. electronic system, voltage source



b. rotational mechanical system, angular velocity source



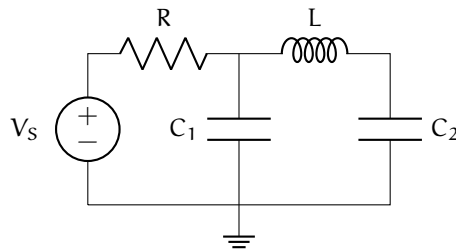
c. translational mechanical system, force source



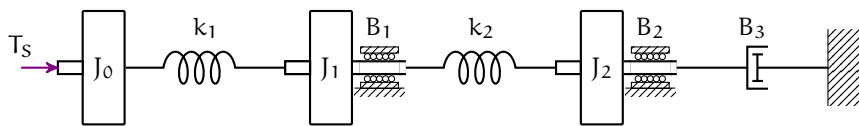
**Exercise 02.7 kurt**

Draw necessary sign coordinate arrows and a linear graph for each of the following schematics.

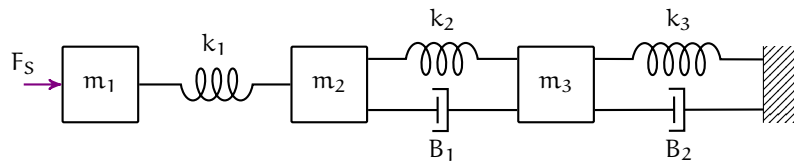
- a. electronic system, voltage source



- b. rotational mechanical system, torque source

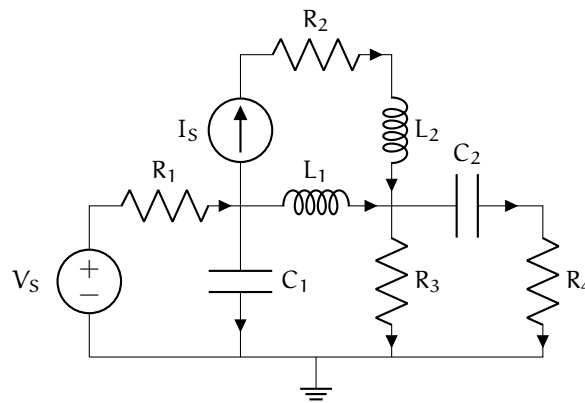


- c. translational mechanical system, force source

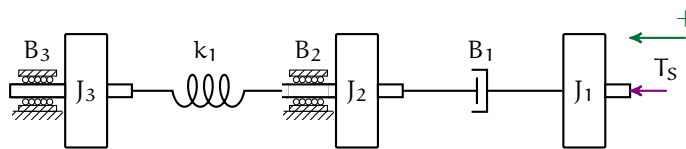
**Exercise 02.8 bunker**

Use the assigned coordinate arrows to draw a linear graph for each of the following schematics.

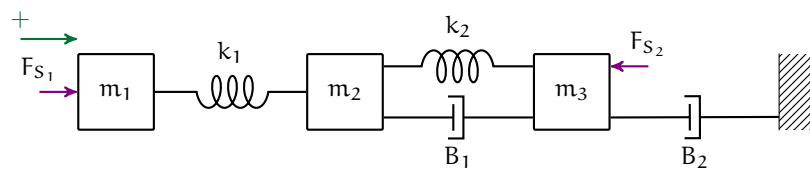
- a. electronic system, voltage and current source



b. rotational mechanical system, torque source, coordinate arrow



c. translational mechanical system, force sources (2)





## 03.1 *ss.svar* State variable system representation

**1 State variables**, typically denoted  $x_i$ , are members of a minimal set of variables that completely expresses the **state** (or status) of a system. All variables in the system can be expressed algebraically in terms of state variables and **input variables**, typically denoted  $u_i$ .

**2 A state-determined system model** is a system for which

1. a mathematical description in terms of  $n$  state variables  $x_i$ ,
2. initial conditions  $x_i(t_0)$ , and
3. inputs  $u_i(t)$  for  $t \geq t_0$

are sufficient conditions to determine  $x_i(t)$  for all  $t \geq t_0$ . We call  $n$  the **system order**.

**3** The state, input, and **output variables** are all functions of time. Typically, we construct vector-valued functions of time for each. The so-called **state vector**  $\mathbf{x}$  is actually a vector-valued function of time  $\mathbf{x} : \mathbb{R} \rightarrow \mathbb{R}^n$ . The  $i$ th value of  $\mathbf{x}$  is a state variable denoted  $x_i$ .

**4** Similarly, the so-called **input vector**  $\mathbf{u}$  is actually a vector-valued function of time  $\mathbf{u} : \mathbb{R} \rightarrow \mathbb{R}^r$ , where  $r$  is the number of *inputs*. The  $i$ th value of  $\mathbf{u}$  is an input variable denoted  $u_i$ .

**5** Finally, the so-called **output vector**  $\mathbf{y}$  is actually a vector-valued function of time  $\mathbf{y} : \mathbb{R} \rightarrow \mathbb{R}^m$ , where  $m$  is the number of *outputs*. The  $i$ th value of  $\mathbf{y}$  is an output variable denoted  $y_i$ .

**6** Most systems encountered in engineering practice can be modeled as state-determined. For these systems, the number of state variables  $n$  is equal to the number of **independent energy storage elements**.

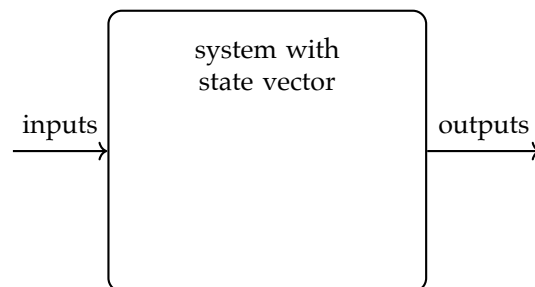
**7** Since to know the state vector  $\mathbf{x}$  is to know everything about the state, the energy stored in each element can be determined from  $\mathbf{x}$ . Therefore, the time-derivative  $d\mathbf{x}/dt$  describes the **power flow**.

**8** The choice of state variables represented by  $\mathbf{x}$  is not unique. In fact, any basis transformation yields another valid state vector. This is because, despite a vector's *components* changing when its basis is changed, a



“symmetric” change also occurs to its *basis vectors*. This means *a vector is a coordinate-independent object*, and the same goes for vector-valued functions. This is not to say that there aren’t invalid choices for a state vector. There are. But if a valid state vector is given in one basis, any basis transformation yields a valid state vector.

9 One aspect of the state vector *is* invariant, however: it must always be a vector-valued function in  $\mathbb{R}^n$ . Our method of analysis will yield a special basis for our state vectors. Some methods yield rather unnatural state variables (e.g. the third time-derivative of the voltage across a capacitor), but ours will yield natural state variables (e.g. the voltage across a capacitor or the force through a spring).



**Figure svar.1:** block diagram of a system with input  $u$ , state  $x$ , and output  $y$ .

## 03.2 ss.ssmode1 State and output equations

1 The state  $\mathbf{x}$ , input  $\mathbf{u}$ , and output  $\mathbf{y}$  vectors interact through two equations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (1a)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \quad (1b)$$

where  $\mathbf{f}$  and  $\mathbf{g}$  are vector-valued functions that depend on the system. Together, they comprise what is called a **state-space model** of a system. Let's not glide past these equations, which will be our dear friends for the rest of our analytic lives. The first equation (1a) is called the **state equation**. Given state and input vectors at a moment in time, it's function  $\mathbf{f}$  describes, *how the state is changing* (i.e.  $d\mathbf{x}/dt$ ). Clearly, the state equation is a vector differential equation, which is equivalent to a system of first-order differential equations.<sup>1</sup>

2 In accordance with the definition of a state-determined system from [Lecture 03.1 ss.svar](#), given an initial condition  $\mathbf{x}(t_0)$  and input  $\mathbf{u}$ , the state  $\mathbf{x}$  is determined for all  $t \geq t_0$ . The state-space model is precisely the "mathematical model" described in the definition of a state-determined system. Determining the state requires the solution—analytic or numerical—of the vector differential equation.

3 The second equation (1b) is *algebraic*. It expresses how the output  $\mathbf{y}$  can be constructed from the state  $\mathbf{x}$  and input  $\mathbf{u}$ . This means we must first solve the state equation (1a). Since the output  $\mathbf{y}$  is a vector of variables of interest, the output equation is constructed in two steps: (1) define the output variables and (2) write them in terms of the state variables  $x_i$  and input variables  $u_j$ .

4 Just because we know that, for a state-determined system, there exists a solution to [Equation 1a](#), doesn't mean we know how to find it. In general,  $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^n$  and  $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^m$  can be nonlinear functions.<sup>2</sup>

<sup>1</sup>We'll learn how to solve such systems both analytically and numerically in later chapters.

<sup>2</sup>Technically, since  $\mathbf{x}$  and  $\mathbf{u}$  are themselves functions,  $\mathbf{f}$  and  $\mathbf{g}$  are *functionals*.

We don't know how to solve most nonlinear state equations analytically. An additional complication can arise when, in addition to states and inputs, system parameters are themselves time-varying (note the explicit time  $t$  argument of  $f$  and  $g$ ). Fortunately, often a linear model is sufficiently effective.<sup>3</sup>

**5 A linear, time-invariant (LTI) system has state-space model**

$$\frac{dx}{dt} = Ax + Bu \quad (2a)$$

$$y = Cx + Du \quad (2b)$$

where

- $A$  is an  $n \times n$  matrix that describes how the state  $x$  changes the state  $x$ ,
- $B$  is an  $n \times m$  matrix that describes how the input  $u$  changes the state  $x$ ,
- $C$  is an  $p \times n$  matrix that describes how the state  $x$  contributes to the output  $y$ , and
- $D$  is an  $p \times m$  matrix that describes how the input  $u$  contributes to the output  $y$ .

In the next two lectures, we will learn how to derive a state-space model—for linear systems, how to find  $A$ ,  $B$ ,  $C$ , and  $D$ —for a system *from its linear graph*. This is the link between the linear graph model and the state-space model.

---

<sup>3</sup>A later lecture will describe the process of deriving a “linearized” model from a nonlinear one.

### 03.3 ss.graph2nt Normal trees

1 Before we introduce the algorithm for constructing the state-space model in [Lecture 03.4 ss.nt2ss](#), we introduce the first step from the system graph to the state-space model: the **normal tree**. It is a **subgraph** of the system's linear graph.

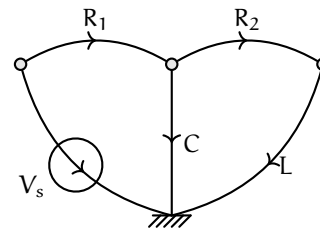
2 In the following, we will consider a connected graph with  $E$  edges, of which  $S$  are sources. There are  $2E - S$  unknown across- and through-variables, so that's how many equations we need. We have  $E - S$  elemental equations and for the rest we will write continuity and compatibility equations.  $N$  is the number of nodes.

3 The following rules must be respected.

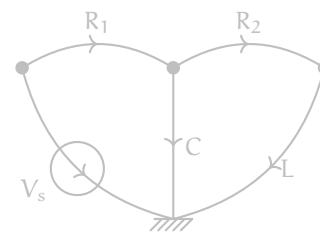
R1. There can be no loops.

R2. Every node must be connected.

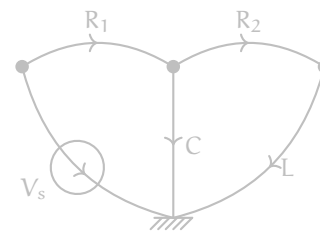
Form a normal tree with the following steps. For an inline example, we will construct a normal tree from the linear graph for an electronic system, shown at right.



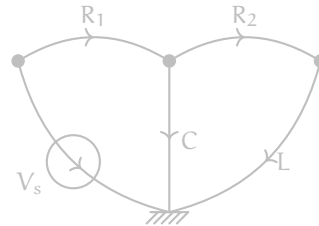
1. Include all nodes.



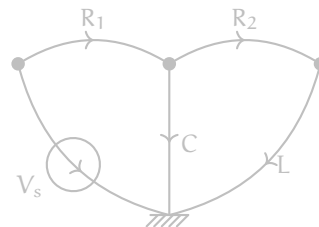
2. Include all across-variable sources.



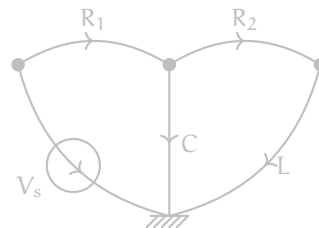
3. Select as many as possible A-type elements.



4. Select as many as possible D-type elements.



5. Select as many as possible T-type elements.



4 We call those edges in the normal tree its **branches** and those not, the **links**.

5 *A-type elements not in* and *T-type elements in* the normal tree are called **dependent energy storage elements**. All other A- and T-types are **independent energy storage elements**. The energy in these can be independently controlled.

6 In order to avoid an artificial excess in state variables and construct what is called a **controllable** model, whenever **A-types in series** (sharing one node) or **T-types in parallel** (sharing two nodes) appear, we should combine them to form equivalent elements in accordance with the formulas

$$C_e = \frac{1}{\sum_i 1/C_i} \quad \text{or} \quad (1a)$$

$$L_e = \frac{1}{\sum_i 1/L_i}. \quad (1b)$$

7 There are special names for power-flow variables associated with an element, depending on whether the element is a branch or link. **Primary**

**variables** are: *across*-variables on branches and *through*-variables on links.

**Secondary variables** are: *through*-variables on branches and *across*-variables on links.

## 03.4 ss.nt2ss Normal tree to state-space

1 At long last, we consider an algorithm to generate a state-space model from a linear graph model. In the following, we will consider a connected graph with  $E$  edges, of which  $S$  are sources (split between through-variable sources  $S_T$  and across  $S_A$ ). There are  $2E - S$  unknown across- and through-variables, so that's how many equations we need. We have  $E - S$  elemental equations and for the rest we will write continuity and compatibility equations.  $N$  is the number of nodes.

1. Derive  $2E - S$  independent differential and algebraic equations from elemental, continuity, and compatibility equations.
  - a) Draw a **normal tree**.
  - b) Identify **primary** and **secondary variables**.
  - c) Select the **state variables** to be
    - across*-variables on A-type branches and
    - through*-variables on T-type links.
  - d) Define the **state vector**  $x$ , **input vector**  $u$ , and **output vector**  $y$ .
  - e) Write an **elemental equation** for each passive element.<sup>4</sup>
  - f) Write a **continuity equation** for each passive branch by drawing a contour intersecting that and no other branch. Solve each for the secondary through-variable associated with that branch.<sup>5</sup>
  - g) Write a **compatibility equation** for each passive link by temporarily "including" it in the tree and finding the compatibility equation for the resulting loop. Solve each for the secondary across-variable associated with that link.<sup>6</sup>
2. Eliminate variables that are not state or input variables and their derivatives. The following procedure is recommended.

<sup>4</sup>There will be  $E - S$  elemental equations.

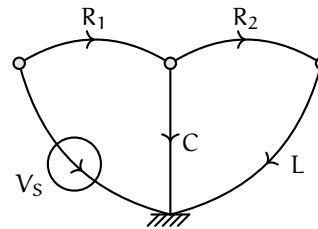
<sup>5</sup>There will be  $N - 1 - S_A$  independent continuity equations.

<sup>6</sup>There will be  $E - N + 1 - S_T$  independent compatibility equations.

- a) Eliminate all secondary variables by substitution into the elemental equations of the continuity and compatibility equations.
- b) Reduce the resulting set of equations to  $n$  (system order) in state and input variables, only. If not elimination, use linear algebra.
- c) Write the result in standard form (Equation 1a or Equation 2a).
- d) Express the output variables in terms of state and input variables, using any of the elemental, continuity, or compatibility equations.
- e) Write the result in standard form (Equation 1b or Equation 2b).

**Example 03.4 ss.nt2ss-1**

For the electronic system shown, find a state-space model with outputs  $i_L$ ,  $I_s$ , and  $v_{R_2}$ .



re:  
circuit  
state-  
space  
model







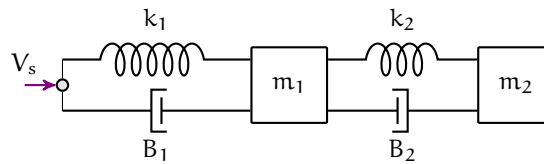


## 03.5 ss.trans State-space model of a translational mechanical system

1 Let's try an example of a higher-order translational mechanical system.

### Example 03.5 ss.trans-1

For the translational mechanical system shown, find a state-space model with outputs the spring forces and mass momenta.



re:  
state-  
space  
model  
of a  
translational  
mechanical  
system

*STATE-SPACE MODEL OF A TRANSLATIONAL MECHANICAL SYSTEM* 92



*STATE-SPACE MODEL OF A TRANSLATIONAL MECHANICAL SYSTEM* 93



*STATE-SPACE MODEL OF A TRANSLATIONAL MECHANICAL SYSTEM* 94

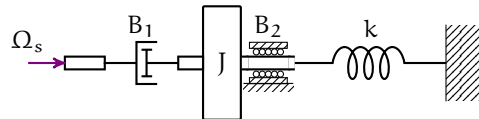


## 03.6 ss.rot State-space model of a rotational mechanical system

1 Let's try an example of a rotational mechanical system.

### Example 03.6 ss.rot-1

For the rotational mechanical system shown, find a state-space model with outputs the spring torque and moment of inertia angular momentum.



re:  
state-  
space  
model  
of a  
rotational  
mechanical  
system









## 03.7 ss.ss2tf2io Bridge between state-space and io differential equations

1 The **Laplace transform**  $\mathcal{L}$  is cool af. It is used to solve differential equations and define the **transfer function**  $H$ : you know, just another awesome dynamic system representation. For now, we'll use it as a bridge between state-space and input/output differential equation representations, merely waving at transfer functions as we pass through. Later, transfer functions will be considered extensively.

### Transfer functions

2 Let a system have an input  $u$  and an output  $y$ . Let the Laplace transform of each be denoted  $U$  and  $Y$ , both functions of complex Laplace transform variable  $s$ . A **transfer function**  $H$  is defined as the ratio of the Laplace transform of the output over the input:

$$H(s) = \frac{Y(s)}{U(s)}. \quad (1)$$

3 The transfer function is exceedingly useful in many types of analysis. One of its most powerful aspects is that it gives us access to thinking about systems as operating on an input  $u$  and yielding an output  $y$ .

### Bridging transfer functions and io differential equations

4 Consider a dynamic system described by the *input-output differential equation*—with variable  $y$  representing the *output*, dependent variable time  $t$ , variable  $u$  representing the *input*, constant coefficients  $a_i, b_j$ , order  $n$ , and  $m \leq n$  for  $n \in \mathbb{N}_0$ —as:

$$\begin{aligned} \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = \\ b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u. \end{aligned} \quad (2)$$

5 The **Laplace transform**  $\mathcal{L}$  of Eq. 2 yields something interesting (assuming zero initial conditions):

$$\begin{aligned} & \mathcal{L} \left( \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y \right) = \\ & \mathcal{L} \left( b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u \right) \Rightarrow \\ & \mathcal{L} \left( \frac{d^n y}{dt^n} \right) + a_{n-1} \mathcal{L} \left( \frac{d^{n-1} y}{dt^{n-1}} \right) + \cdots + a_1 \mathcal{L} \left( \frac{dy}{dt} \right) + a_0 \mathcal{L}(y) = \\ & b_m \mathcal{L} \left( \frac{d^m u}{dt^m} \right) + b_{m-1} \mathcal{L} \left( \frac{d^{m-1} u}{dt^{m-1}} \right) + \cdots + b_1 \mathcal{L} \left( \frac{du}{dt} \right) + b_0 \mathcal{L}(u) \Rightarrow \\ & s^n Y + a_{n-1} s^{n-1} Y + \cdots + a_1 s Y + a_0 Y = \\ & b_m s^m U + b_{m-1} s^{m-1} U + \cdots + b_1 s U + b_0 U. \end{aligned}$$

Solving for  $Y$ ,



The inverse Laplace transform  $\mathcal{L}^{-1}$  of  $Y$  is the **forced response**. However, this is not our primary concern; rather, we are interested to solve for the transfer function  $H$  as the ratio of the output transform  $Y$  to the input transform  $U$ , i.e.

$$H(s) \equiv \frac{Y(s)}{U(s)} \quad (3)$$

$$= \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}. \quad (4)$$

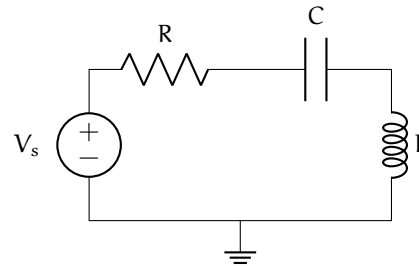
6 Exactly the reverse procedure, then, can be used to convert a given transfer function to an input-output differential equation.

**Example 03.7 ss.ss2tf2io-1**

The circuit shown has input-output differential equation

$$L \frac{d^2 v_L}{dt^2} + R \frac{dv_L}{dt} + \frac{1}{C} v_L = L \frac{d^2 V_s}{dt^2}.$$

What is the transfer function from  $V_s$  to  $v_L$ ?



re: A  
circuit  
transfer  
function

**Bridging transfer functions and state-space models**

7 Given a system in the standard form of a state equation,

$$\frac{dx}{dt} = Ax + Bu,$$

we take the Laplace transform to yield, assuming zero initial conditions,

which can be solved for the state:

(5)

where  $I$  is the identity matrix with the same dimension as that of  $A$ . The standard form of the output equation yields the output solution

$$Y = HU, \tag{6}$$

where we define the **matrix transfer function**  $H$  to be



The element  $H_{ij}$  is a transfer function from the  $j$ th input  $U_j$  to the  $i$ th output  $Y_i$ .

8 The reverse procedure of deriving a state-space model from a transfer function is what is called a **state-space realization**, which is not a unique operation (there are different realizations for a single transfer function) and is not considered here.

### Example 03.7 ss.ss2tf2io-2

Given the linear state-space model

$$\dot{\mathbf{x}} = \begin{bmatrix} -3 & 4 \\ -1 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \mathbf{u},$$

••• derive the matrix transfer function.

re:  
Matrix  
transfer  
function  
from  
state-  
space

**Example 03.7 ss.ss2tf2io-3**

For the following state-space model, derived in Example 03.4 ss.nt2ss-1, derive the io differential equations for each output variable:

$$\frac{dx}{dt} = \begin{bmatrix} -\frac{1}{R_1 C} & -\frac{1}{C} \\ 1/L & -R_2/L \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{1}{R_1 C} \\ 0 \end{bmatrix} \mathbf{u}$$
$$\mathbf{y} = \begin{bmatrix} 0 & 1 \\ -1/R_1 & 0 \\ 0 & R_2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1/R_1 \\ 0 \end{bmatrix} \mathbf{u}.$$

The output variables are  $i_L$ ,  $I_S$ , and  $v_{R_2}$ .

re:  
state-  
space  
to io  
differential  
equations





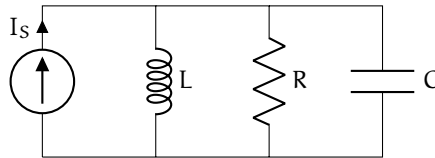
## 03.8 ss.exe Exercises for Chapter 03

### SS

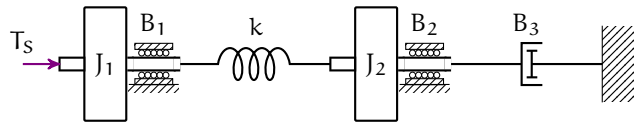
#### Exercise 03.1 metroid

Draw necessary sign coordinate arrows, a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following schematics.

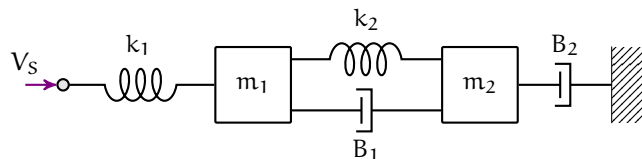
- a. Electrical system, current source



- b. Rotational mechanical system, torque source



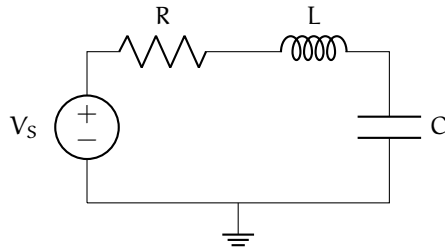
- c. Translational mechanical system, velocity source



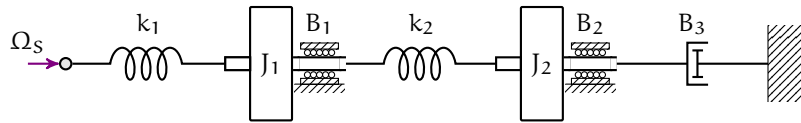
#### Exercise 03.2 megaman

Draw necessary sign coordinate arrows, a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following schematics.

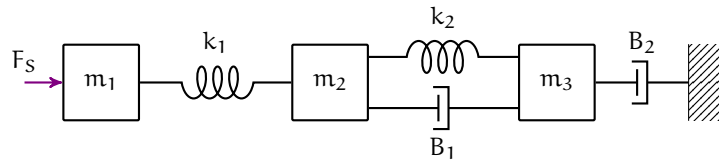
- a. Electrical system, voltage source



b. Rotational mechanical system, angular velocity source



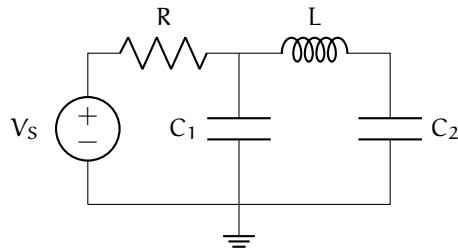
c. Translational mechanical system, force source



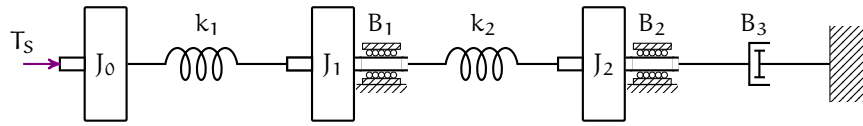
**Exercise 03.3 sonic**

Draw necessary sign coordinate arrows, a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following schematics.

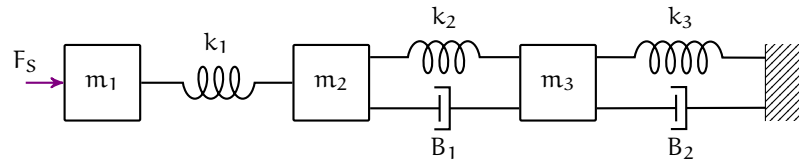
a. Electrical system, voltage source



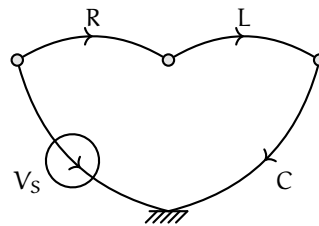
b. Rotational mechanical system, torque source



c. Translational mechanical system, force source

**Exercise 03.4 nintendo**

Use the following linear graph for a circuit to answer the questions below, which are the steps to determining a state-space model of the circuit. Use the sign convention from the diagram.  $V_S$  is a voltage source.

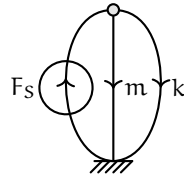


- Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector* for the outputs  $i_R$  and  $v_C$ .
- Write the required *elemental*, *continuity*, and *compatibility equations*.
- Solve for the *state equation* in standard form.
- Solve for the *output equation* in standard form.

**Exercise 03.5 supernintendo**

Use the following linear graph for a mechanical translational system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign convention from the diagram.  $F_S$  is a force source. Let the outputs be  $v_m$  and  $f_k$ .

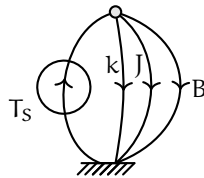


- Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
- Write the required *elemental*, *continuity*, and *compatibility equations*.
- Solve for the *state equation* in standard form.
- Solve for the *output equation* in standard form.

### Exercise 03.6 gameboy

Use the following linear graph for a mechanical rotational system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign convention from the diagram.  $T_S$  is a torque source. Let the outputs be  $\Omega_J$  and  $T_B$ .

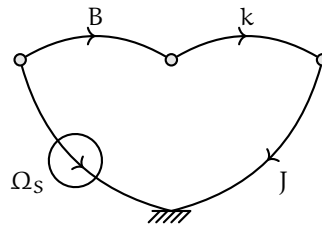


- Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
- Write the required *elemental*, *continuity*, and *compatibility equations*.
- Solve for the *state equation* in standard form.
- Solve for the *output equation* in standard form.

**Exercise 03.7 blowhard**

Use the following linear graph for a mechanical rotational system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign convention from the diagram.  $\Omega_S$  is an angular velocity source. Let the outputs be the angular velocity  $\Omega_J$  of the inertia and the angular displacement  $\theta_k$  across the spring.

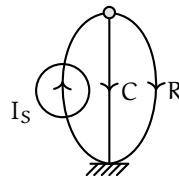


1. Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
2. Write the required *elemental*, *continuity*, and *compatibility equations*.
3. Solve for the *state equation* in standard form.
4. Solve for the *output equation* in standard form.

**Exercise 03.8 blinken**

Use the following linear graph for an electrical system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign assignments from the diagram.  $I_S$  is a current source. Let the outputs be the voltage across the capacitor  $v_C$  and the current through the resistor  $i_R$ .

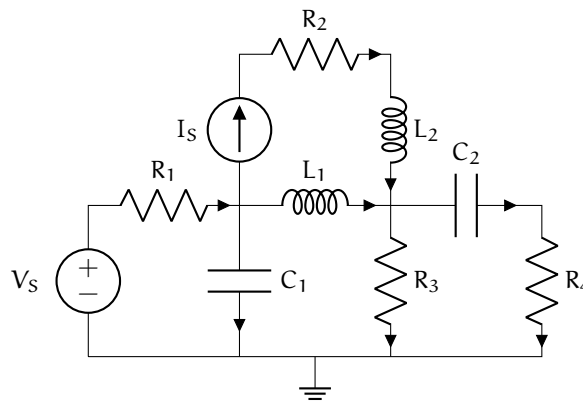


1. Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
2. Write the required *elemental*, *continuity*, and *compatibility equations*.
3. Solve for the *state equation* in standard form.
4. Solve for the *output equation* in standard form.

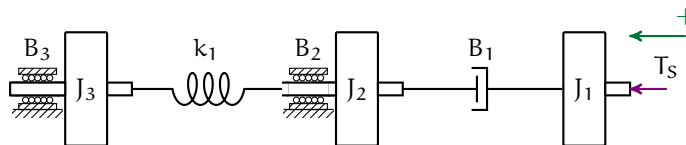
### Exercise 03.9 chunker

Use the assigned coordinate arrows to draw a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following schematics.

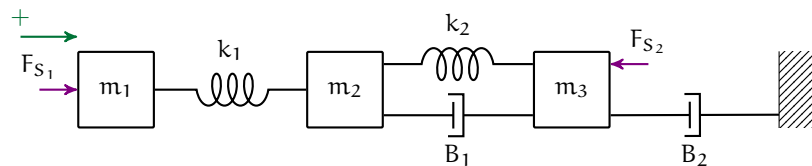
1. Electrical system, voltage and current source



2. Rotational mechanical system, torque source, coordinate arrow



3. Translational mechanical system, force sources (2)

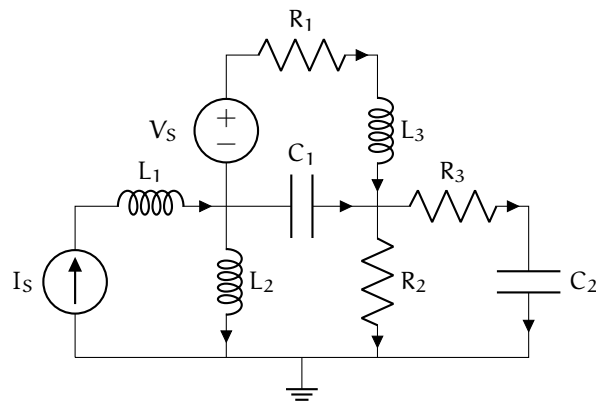


**Exercise 03.10 stevenuniverse**

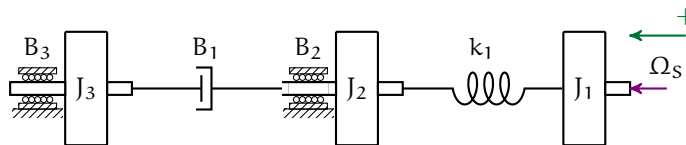
Use the assigned coordinate arrows to draw a *linear graph*, a *normal tree*, and identify *state variables*, *system order*, and *dependent energy storage elements* for each of the following schematics.

\_\_\_\_\_/ 30 p.

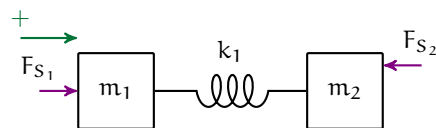
1. Electrical system, voltage and current source



2. Rotational mechanical system, angular velocity source



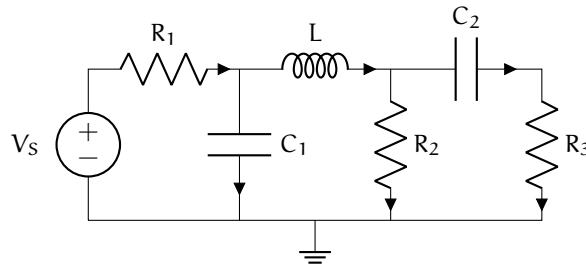
3. Translational mechanical system, force sources (2)



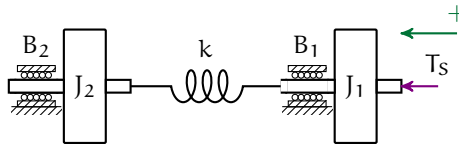
**Exercise 03.11 winken**

Use the assigned coordinate arrows to draw a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following schematics.

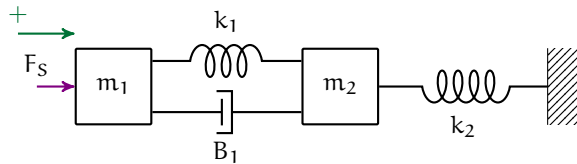
1. Electrical system, voltage source



2. Rotational mechanical system, torque source, coordinate arrow



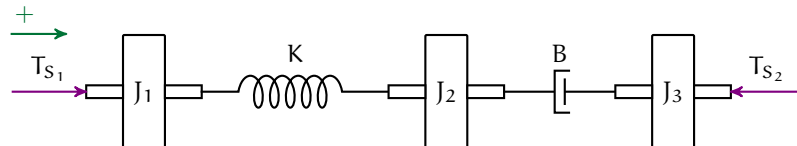
3. Translational mechanical system, force source, coordinate arrow



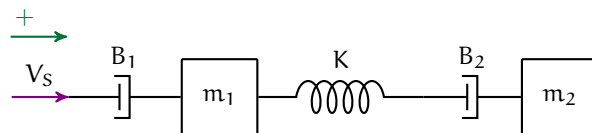
**Exercise 03.12 granada**

Use the assigned coordinate arrows to draw a *linear graph*, a *normal tree*, and identify *state variables* and *system order* for each of the following systems.

1. Rotational mechanical system, two torque sources



2. Translational mechanical system, velocity source

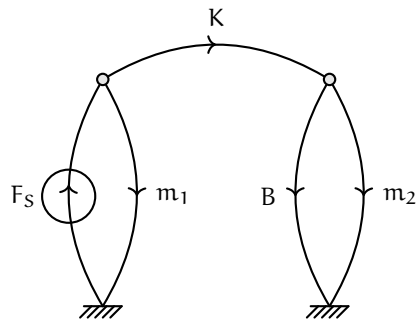




**Exercise 03.13 valencia**

Use the following linear graph for a mechanical translational system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign convention from the diagram.  $F_S$  is a force source. Let the outputs be  $v_{m_1}$  and  $v_{m_2}$ .



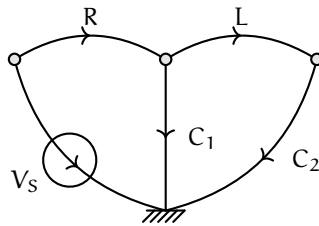
1. Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
2. Write the required *elemental*, *continuity*, and *compatibility equations*.
3. Solve for the *state equation* in standard form.
4. Solve for the *output equation* in standard form.

**Exercise 03.14 stevenash**

Use the linear graph of Fig. exe.1 for an electrical system to answer the questions below, which are the steps to determining a state-space model from the linear graph.

Use the sign convention from the diagram.  $V_S$  is a voltage source. Let the outputs be  $v_{C_2}$ ,  $v_R$ , and  $i_S$  (i.e., the source current).

\_\_\_\_\_/   
 35 p.



**Figure exe.1:** A linear graph of an electrical system.

1. Determine the *normal tree*, *state variables*, *system order*, *state vector*, *input vector*, and *output vector*.
2. Write the required *elemental*, *continuity*, and *compatibility equations*.
3. Solve for the *state equation* in standard form.
4. Solve for the *output equation* in standard form.

## 04 emech

---

## 04.1 emech.trans Ideal transducers

- 1 **Two-port** system elements can model **transducers**—elements that transfer energy between two energy domains or change its form within an energy domain. The quintessential example, which we will consider in detail, is the **motor**, which converts electrical energy to mechanical energy. However, many other system elements can be considered transducers, and we'll consider a few in this lecture.
- 2 Each of the two ports has a through- and an across-variable. We use the convention that the power *into* each port ( $\mathcal{P}_1$  and  $\mathcal{P}_2$ ) is positive, which has implications for the signs of the power flow variables  $\mathcal{F}_1$ ,  $\mathcal{F}_2$ ,  $\mathcal{V}_1$ , and  $\mathcal{V}_2$ . For an two-port element to transfer power, we have



We define the **transformer ratio** TF to be

$$\text{TF} \equiv \frac{\mathcal{V}_1}{\mathcal{V}_2} = -\frac{\mathcal{F}_2}{\mathcal{F}_1}. \quad (1)$$

Furthermore, we define the **gyrator modulus** GY to be

$$\text{GY} \equiv \frac{\mathcal{V}_1}{\mathcal{F}_2} = -\frac{\mathcal{V}_2}{\mathcal{F}_1}. \quad (2)$$

- 3 For an **ideal transducer**—one that is linear, time-invariant, and without power loss—we have only two nontrivial solutions:<sup>1</sup>

$$\begin{array}{ccc} \mathcal{V}_2 = \mathcal{V}_1/\text{TF} & \text{or} & \mathcal{V}_2 = -\text{GY} \mathcal{F}_1 \\ \mathcal{F}_2 = -\text{TF} \mathcal{F}_1 & & \mathcal{F}_2 = \mathcal{V}_1/\text{GY}. \end{array}$$

- 4 For a given element, if the solution with TF is a good model, we call that element a **transformer**. If the GY solution is a good model, we call it a **gyrator**.

<sup>1</sup>For an explanation of *why* that is the case, see Rowell and Wormley (1997).

**Example 04.1 emech.trans-1****re: DC  
motor**

Consider a DC motor with rotor radius  $r$ , number of coil turns  $N$ , background field  $B$ , and rotor length  $\ell$ . The torque  $T$  of a DC motor is related to its coil current  $i$  by the relation

$$T = -2rNB\ell i.$$

1. Determine if DC motors are transformers or gyrators.
2. Find TF or GY.
3. Derive the relation between the voltage  $v$  and the angular velocity  $\Omega$  across the motor using the assumption that it is an ideal transducer.

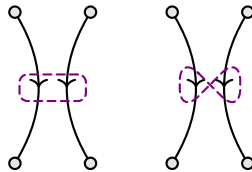
**Example 04.1 emech.trans-2****re:  
gears**

Consider two gears with radii  $r_1$  and  $r_2$  and number of teeth  $n_1$  and  $n_2$ .

1. Determine the power flow variables for gears.
2. Write two independent equations relating the power flow variables.
3. Determine if gears are transformers or gyrators.
4. Find TF or GY.



## 04.2 emech.transmod Modeling with transducers



**Figure transmod.1:** two-port ideal linear graph elements of a transformer (left) and a gyrator (right).

**1** We now develop both linear graph and state-space models of systems that include transducers. Linear graphs of two-port ideal transducer elements are drawn as shown in [Figure transmod.1](#). Once again, we use the sign convention that power into an element is positive. Often, the edges are drawn toward ground nodes, which are always different when the transducer acts between different energy domains. Transducers may or may not be sufficiently modeled by ideal transducers. For instance, we may need to consider the moment of inertia associated with a gear. When this is the case, additional elements can be connected in parallel and in series with the two-port element nodes. DC motors—another example—are typically not modeled with an ideal transducer, alone, because the windings have both resistance and inductance.

### State-space modeling with transducers

**2** We present a method for constructing a state-space model of systems containing transducer elements. This procedure begins, as before, with the construction of the normal tree. The following rules must be respected.

- R1. There can be no loops.
- R2. Every node must be connected.
- R3. Of a transformer's two edges, exactly one is included.
- R4. Of a gyrator's two edges, either both are or neither is included.

- 3 Form a normal tree with the following steps.
  1. Include all nodes.
  2. Include all across-variable sources.
  3. Include as many as possible A-type elements.<sup>2</sup>
  4. Include transducer edges, minimizing the number of T-types in the tree.
  5. Include as many as possible D-type elements.
  6. Include as many as possible T-type elements.
- 4 The state and output equations can be derived as before, but with the following caveat: each two-port element requires two elemental equations.

---

<sup>2</sup>Inclusion of an A-type at this step may result in a violation of R3 or R4 in the next, which implies the A-type is a *dependent energy storage element* and that it should be excluded from the normal tree.



## 04.3 emech.dcm DC motors

- 1 DC motors are commonly used in mechanical engineering designs as an actuator. Products such as pumps, fans, conveyors, and robots use DC motors to convert electrical energy to mechanical (rotational) energy.
- 2 DC motors first emerged in the mid-19th century as the first device to produce useful mechanical work from electrical power.<sup>3</sup> One of fathers of the DC motor, the Benedictine priest *Ányos Jedlik*, invented the key facets of the motor: the **stator**, the **rotor**, and the **commutator**. Roughly speaking, for a typical brushed DC motor, current flowing through the wire windings of the stator produces a magnetic field that turns the rotor, which has windings of its own; the commutator mechanically switches the direction of current flow through the windings to yield continuous electromagnetic torque.
- 3 We will begin our study of DC motors with a review of a key physical phenomenon: the mechanical force on a charged particle moving in a magnetic field.

### Lorentz force

- 4 Consider a charged particle moving through a background magnetic field. The **Lorentz force** is the (mechanical!) force on the particle, which depends on the velocity of the particle, the background magnetic field, and the background electric field. Charge flowing through a straight, stationary<sup>4</sup> wire with current  $i$  in a uniform background magnetic field  $\mathbf{B}$  is subject to the cumulative effect of the Lorentz force on each charge. Let the straight wire's length and orientation in the B-field be described by the vector  $\ell$ , which should be chosen to be in the direction of positive current flow. It can be shown that the resultant force  $\mathbf{f}$  on the wire is

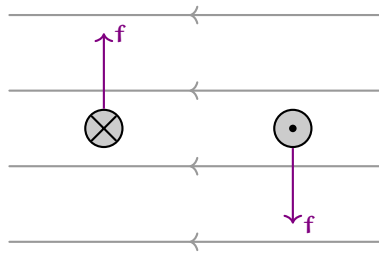
$$\mathbf{f} = i\ell \times \mathbf{B} \quad (1)$$

as shown in [Fig. dcm.1](#).

---

<sup>3</sup>See a decent history [here](#).

<sup>4</sup>The equations here assume a stationary wire. In a DC motor, the wire is moving, which creates additional effects, but the Lorentz force is still present.



**Figure dcm.1:** the forces  $f$  on two wires in a magnetic field  $B$ . The wire on the left has current flowing *into* the board, that on the right has current flowing *out* of the board. The cross-product right-hand-rule applies.

5 With a curved wire, then, we could take infinitesimal sections  $d\ell$  and integrate along the wire's path:

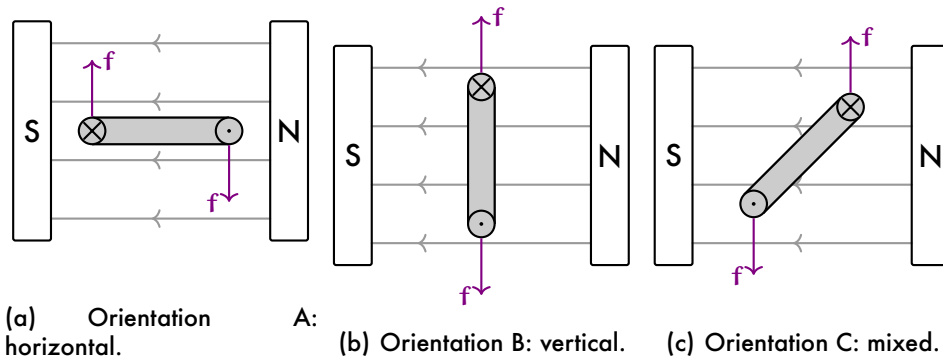
$$\mathbf{f} = i \int d\boldsymbol{\ell} \times \mathbf{B}. \quad (2)$$

6 DC motors take advantage of this electromechanical phenomenon by driving current through cleverly arranged wires to generate torque on a shaft.

### Permanent magnet DC motors

7 In order to take advantage of the Lorentz force, first a uniform background magnetic field  $\mathbf{B}$  is required. Some DC motors, called **permanent magnet DC motors** (PMDC motors) generate this field with two stationary permanent magnets arranged as shown in Fig. dcm.2. The magnets are affixed to the “stationary” part of the motor called the **stator**.

8 Now consider a rigidly supported wire with current  $i$  passing through the field such that much of its length is perpendicular to the magnetic field. Consider the resultant forces on these perpendicular sections of wire for different wire configurations, as illustrated in Fig. dcm.2. We have torque! But note that it changes direction for different armature orientations, which will need to be addressed in a moment. Note that we can wind this wire—which we call the **armature**—multiple times around the loop to increase the torque. The rotating bit of the motor that supports the armature is called the **rotor**, which includes the shaft.



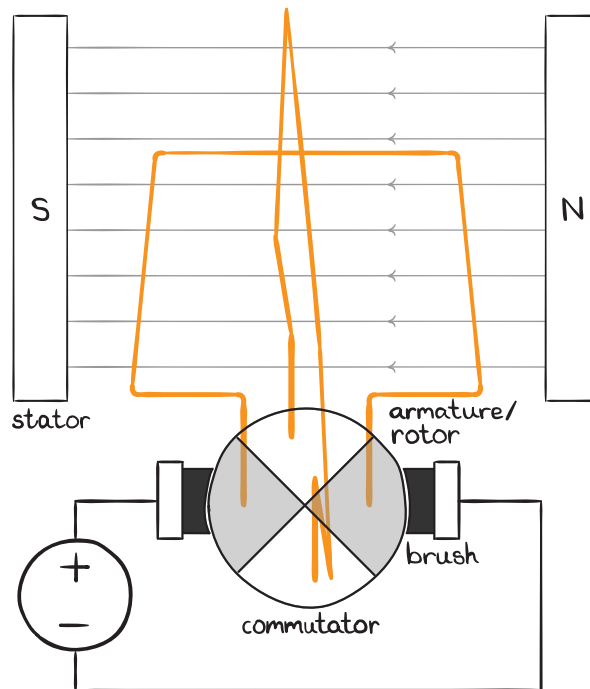
**Figure dcm.2:** axial section view of a simple DC motor with permanent magnets.

9 The trouble is, if we connect our armature up to a circuit—which is usually located alongside the stator, i.e. *not rotating*—the wire will wrap about itself, which is **not # winning**. But we're tricky af so let's consider just cutting that wire and rigidly connecting it to a disk—called a **commutator**—with two conductive regions, one for each terminal of the armature. The commutator will rotate with the armature, but it provides smooth contacts along the perimeter of the disk.

10 We can then connect the driving circuit to these contacts via **brushes**: conductive blocks pressed against the commutator on opposite sides such that they remain in contact (conducting current) yet allow the commutator to slide easily, as shown in Fig. dcm.3. Brushes are typically made from carbon and wear out over time. This is partially mitigated by spring-loading, but eventually the brushes must be replaced, nonetheless.

11 So brushes solve the “wire wrapping” problem, but do they have an effect on the “torque flipping” issue? Yes! When the armature passes through its vertical orientation, *current reverses direction through the armature*. So whenever the perpendicular section of wire is on the right, current flows in the same direction, regardless of to which side of the armature it belongs.

12 Finally, is there a way to overcome the limitation of torque variation with different armature angles? Yes: if there are several different armature windings at different angles and correspondingly the commutator is split



**Figure dcm.3:** illustration of brushes, commutator, and two armatures.

into several conductive contact pairs (one for each armature winding), a relatively continuous torque results! Real PMDC motors use this technique.

### Wound stator DC motors

**13** **Wound stator DC motors** operate very similarly to PMDC motors, but generate their background field with two stationary coils in place of the permanent magnets, above. These electromagnets require a current of their own, which is usually provided through the same circuitry that supplies the armature current (DC motors typically have only two terminals).

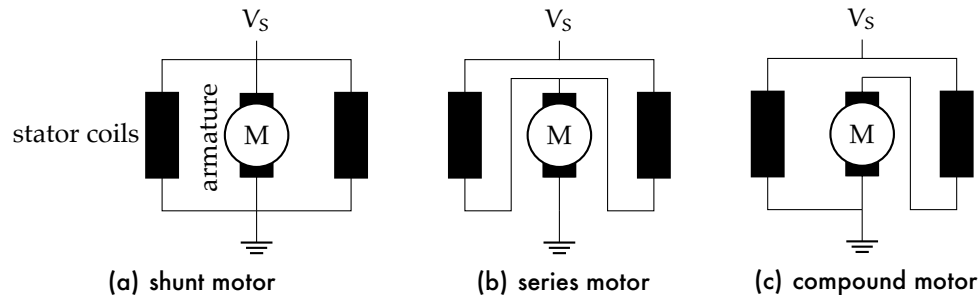
**14** Three common configurations of the electrical connection of these are shown in Fig. dcm.4. These define the following three DC motor types.

**shunt** The **shunt DC motor** has its stator and rotor windings connected in parallel. These are the most common wound stator DC motors and

their speeds can be easily controlled without feedback, but they have very low starting torque.

**series** The **series DC motor** has stator and rotor windings connected in series. These have high starting torque—so high, in fact, that it is not advisable to start these motors without a load—but their speeds are not as easily controlled without feedback.

**compound** The **compound DC motor** has stator and rotor windings connected in both series and parallel. These can exhibit characteristics that mix advantages and disadvantages of shunt and series DC motors.



**Figure dcm.4:** connections for shunt, series, and compound DC motors.

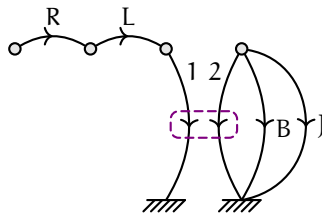
### Brushless DC motors

**15** There is yet another type of DC motor: **brushless** (BLDC). Brushless DC motors work on principles more similar to AC motors, but require complex solid-state switching that must be precisely timed. As their name implies, these motors do not require brushes. A brushless DC motor mathematical model is not presented here, but a nice introduction is given by Baldursson (2005).

**16** The brushed DC motor is still widely used, despite its limitations, which include relatively frequent maintenance to replace brushes that wear out or clean/replace commutators. Other disadvantages of brushed DC motors include their relatively large size, relatively large rotor inertia, heat generated by the windings of the stator and/or rotor, and arcing that creates

electronic interference for nearby electronics. Reasons they are still widely used include that they are inexpensive (about half the cost of brushless DC motors), don't require (but often still use) complex driving circuits, are easy to model, and are easily driven at different speeds; for these reasons, an additional reason emerges: they're relatively easy to design with!

### A PMDC motor model



**Figure dcm.5:** a better brushed DC motor model.

**17** We have already explored a model for a PMDC motor in [Example 04.1 emech.trans-1](#), which yielded elemental equations

$$T_2 = -TFi_1 \quad \text{and} \quad (3)$$

$$\Omega_2 = v_1/TF, \quad (4)$$

where TF is the motor constant. That model assumed neither armature resistance nor inductance were present—that is, it was an ideal transformer model. A linear graph of a much better model for a DC motor is shown in [Figure dcm.5](#). This model includes a resistor R and inductor L in series with an ideal transducer. On the mechanical side, the rotor inertia J and internal bearing damping B are included. The tail ends of R and 2 should be connected to external electrical and mechanical subgraphs, respectively.

### Motor constants

**18** The **motor torque constant**  $K_t$  and **back-emf voltage constant**  $K_v$  are related to the transformer ratio TF derived above to characterize a brushed DC motor's response. If expressed in a set of consistent units—say, SI

units— $K_t$  and  $K_v$  have the same numerical value and are equivalent to TF. Precisely, with consistent units,  $TF = K_v = K_t$ .

**19** However, manufacturers usually use weird units like oz-in/A and V/krpm. If they are given in anything but SI units, we recommend converting to SI for analysis.

**20** Once in SI, we will have something like (for  $x \in \mathbb{R}$ ):



**21** So if we are given either  $K_t$  or  $K_v$ , the unknown constant can be found (in SI units) by converting the known constant to SI.<sup>5</sup>

## Animations

**22** There are some great animations of DC motor operating principles and construction. I've included the url of my favorite, along with some bonus animations for other important types of motors we don't have time to discuss, here.

- Brushed DC motors: [youtu.be/LAtPHANefQo](https://youtu.be/LAtPHANefQo)
- Brushless DC motors: [youtu.be/bCEi0nuODac](https://youtu.be/bCEi0nuODac)
- AC (asynchronous) induction motors: [youtu.be/AQqyGNOP\\_3o](https://youtu.be/AQqyGNOP_3o)
- AC synchronous motors: [youtu.be/Vk2jDXxZIhs](https://youtu.be/Vk2jDXxZIhs)
- Stepper motors: [youtu.be/eyqwLiowZiU](https://youtu.be/eyqwLiowZiU)

---

<sup>5</sup>One more note. When given a torque constant, the unit "oz" means "ounce-force," which is the mass in regular (mass) ounces multiplied by the gravitational acceleration  $g$ .

## 04.4 emech.real Modeling a real electromechanical system



**Figure real.1:** electromechanical systems from the lab.

**1** We now model the electromechanical systems from the laboratory, shown in [Figure real.1](#). The system includes a brushed DC motor (*Electrocraft 23SMDC-LCSS servomotor from Servo Systems*), two shafts, a shaft coupler, two bearings, and a flywheel. The motor's datasheet specifications are given in [Table real.1](#). The mechanical subsystem's inertia is dominated by the stainless steel flywheel with  $J_f = 0.324 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$ . The bearing damping  $B_b$  is the most difficult parameter to determine. Let's begin with the assumption that the combined bearing damping is  $B_b = 20 \cdot 10^{-6} \text{ N}\cdot\text{m}/(\text{rad}/\text{s})$ .

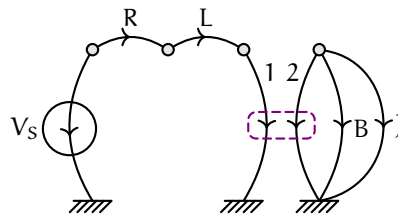
<sup>6</sup>Load applied at one inch from bearing.



**Table real.1:** datasheet specifications for the *Electrocraft 23SMDC-ICSS* servomotor from *Servo Systems*. This is the motor used in the lab.

	parameter	specification	SI conversion
general	continuous stall torque	55 oz-in	0.388 N-m
	peak torque $T_{max}$	400 oz-in	2.82 N-m
	max terminal voltage	60 V <sub>dc</sub>	60 V <sub>dc</sub>
	max operating speed $\Omega_{max}$	6000 rpm	628 rad/s
mechanical	rotor inertia $J_m$	0.008 oz-in/s <sup>2</sup>	$56.5 \cdot 10^{-6}$ N-m/s <sup>2</sup>
	damping constant $B_m$	0.25 oz-in/krpm	$16.9 \cdot 10^{-6}$ N-m/(rad/s)
	thermal resistance	4 C/W	4 K/W
	max armature temp	155 C	428 K
	max friction torque	3 oz-in	0.0212 N-m
	max radial load <sup>6</sup>	10 lb	44.5 N
	weight (motor only)	3.5 lb	15.6 N
electrical	torque constant $K_t$	13.7 oz-in/A	0.097 N-m/A
	voltage constant $K_v$	10.2 V/krpm	0.097 V/(rad/s)
	terminal resistance	1.6 $\Omega$	1.6 $\Omega$
	electrical time constant	2.6 ms	$2.6 \cdot 10^{-3}$ s
	mechanical time constant	8.9 ms	$8.9 \cdot 10^{-3}$ s
	max continuous current	4 A	4 A
	armature inductance	4.1 mH	$4.1 \cdot 10^{-3}$ H
max peak current	34 A	34 A	

**Linear graph model**



**Figure real.2:** a linear graph model of the electromechanical systems of [Figure real.1](#).

2 A linear graph model is in order. An ideal voltage source drives the

motor<sup>7</sup>—modeled as an ideal transducer with armature resistance  $R$  and inductance  $L$ , given in [Table real.1](#). The ideal transducer's rotational mechanical side (2) is connected to a moment of inertia  $J = J_m + J_f = 0.381 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$ , dominated by the flywheel,<sup>8</sup> and damping  $B$ , which is the parallel combination of the internal motor damping of [Table real.1](#) and the bearing damping  $B_b$ , to yield  $B = 26.9 \cdot 10^{-6} \text{ N}\cdot\text{m}/\text{s}^2$ . We choose to ignore the flexibility of the coupler. [Problem 04.4 emech.](#) considers the same system but does not ignore the coupler's flexibility. In general, shaft couplers have significant flexibility and, depending on the application, this may require consideration in the dynamic model.

### State-space model

**3** The normal tree can be constructed by the procedure from [Lecture 04.2 emech.transmod](#). The voltage source  $V_s$  is first included, followed by  $J$ . Then exactly one edge of the ideal transducer must be selected, minimizing the number of T-types in the tree. We don't really have a choice, in this case, because selecting edge 2 would create a loop, so we must select edge 1. Next,  $R$  is included. No more elements can be included without creating a loop, so we are finished.

**4** We are now prepared to determine variables. The state variables are across variables of A-type tree branches and through variables of T-type links—so  $\Omega_J$  and  $i_L$ , and the system is second-order ( $n = 2$ ). Clearly, the system's input is the voltage source  $V_s$ . We are interested in all the variables for the analysis in [Lecture 04.5 emech.curves](#), so we choose them all for our outputs. In summary, then, the state, input, and output vectors are:

$$\mathbf{x} = \begin{bmatrix} \Omega_J \\ i_L \end{bmatrix}, \quad \mathbf{u} = [V_s], \quad \text{and}$$

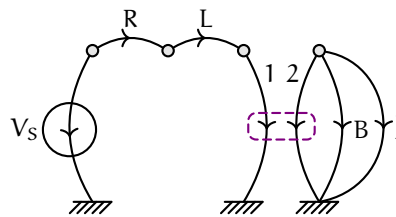
$$\mathbf{y} = [\Omega_J \quad T_J \quad v_L \quad i_L \quad \Omega_B \quad T_B \quad v_R \quad i_R \quad v_1 \quad i_1 \quad \Omega_2 \quad T_2 \quad V_s \quad I_s]^T.$$

<sup>7</sup>Often we can model our motor-driving source as ideal within an operating range. See [Lecture 04.7 emech.drive](#) for more details.

<sup>8</sup>This is the sum of the inertia of the flywheel  $J_f = 0.324 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$  and the rotor  $J_m = 0.0565 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$ . It might be worthwhile combining this with the inertia from the shaft and coupler to obtain a more accurate value, but the difference is likely negligible.

5 Let's write some equations! Elemental are up first.

J	R
L	1
B	2



**Figure real.3:** the linear graph model for drawing contours.

Now, continuity and compatibility equations are developed by summing through-variables into contours. The three required contours—one for each of R, 1, and J—can be drawn on [Figure real.3](#). The three compatibility equations—one for each of L, 2, and B—are found by “temporarily including” those links in the tree and summing across-variables around the loops created. Let's write the equations.

branch	continuity equation	link	compatibility equation
R		L	
1		2	
J		B	

6 All that remains to form the state-space model is to eliminate variables that are neither states nor inputs from the elemental, continuity, and compatibility equations. Eliminating secondary variables by substituting

the continuity and compatibility equations into the elemental equations, the following results.

$$\begin{array}{c|c} J & R \\ \hline L & 1 \\ \hline B & 2 \end{array}$$

The last four equations allow us to eliminate the remaining undesirable variables to obtain the state model in the standard form<sup>9</sup>

$$\frac{dx}{dt} = Ax + Bu \tag{1a}$$

$$y = Cx + Du \tag{1b}$$

where

$$A = \begin{bmatrix} -B/J & TF/J \\ -TF/L & -R/L \end{bmatrix}, \tag{1c}$$

$$B = \begin{bmatrix} 0 \\ 1/L \end{bmatrix}, \tag{1d}$$

$$C = \begin{bmatrix} 1 & -B & -TF & 0 & 1 & B & 0 & 0 & TF & 0 & 1 & 0 & 0 & 0 \\ 0 & TF & -R & 1 & 0 & 0 & R & 1 & 0 & 1 & 0 & -TF & 0 & 1 \end{bmatrix}^T, \text{ and} \tag{1e}$$

$$D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T. \tag{1f}$$

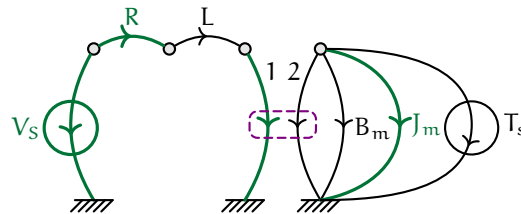
---

<sup>9</sup>Here is the `rnd` file for use with StateMint ([statemint.stmartin.edu](http://statemint.stmartin.edu)) to derive the state-space model from the elemental, continuity, and compatibility equations:

```
ricopic.one/dynamic_systems/source/motor_model.rnd
```

Note that the “constraint equations” are the continuity and compatibility equations solved for primary variables.

## 04.5 emech.curves DC motor performance in steady-state



**Figure curves.1:** a linear graph model of the motor from [Lecture 04.4 emech.real](#) in a test-configuration with a brake modeled by  $T_s$ .

- 1 Brushed DC motor performance has several aspects, but most of them revolve around the so-called **motor curve**: for a given motor voltage, its steady-state speed versus a constant torque applied to the load. The test setup for drawing such a curve requires a calibrated, controllable torque source applied to the motor shaft. A **brake** is typically used. A voltage-controlled **magnetic particle brake** is ideal.<sup>10</sup>
- 2 We will gain a deep understanding of DC motor performance characteristics only by tarrying with this situation. Therefore, we begin by modeling it in [Lecture 04.5 emech.curves](#) and analyzing its performance in [Lecture 04.5 emech.curves](#).

### Modeling the test system

- 3 Including a torque source  $T_s$  on the load changes the model only slightly, as shown in [Figure curves.1](#). Note that the mechanical subsystem is reduced to only the motor, since during such a test the load and bearings would be detrimental (it is a test for the *motor*, after all). Invariant are the normal tree, state variables, and most of the derivation of the state equations.

<sup>10</sup>See, for instance [here](#) or [here](#).

4 The input vector becomes

$$\mathbf{u} = \begin{bmatrix} V_s \\ I_s \end{bmatrix}. \quad (1)$$

The continuity equation for the inertia becomes  $T_{J_m} = -T_2 - T_{B_m} - T_s$  (the torque specifically *opposes* motion, to which we assign the positive direction) and the state model's matrices B and D change, such that<sup>11</sup>

$$A = \begin{bmatrix} -B_m/J_m & TF/J_m \\ -TF/L & -R/L \end{bmatrix}, \quad (2a)$$

$$B = \begin{bmatrix} 0 & -1/J_m \\ 1/L & 0 \end{bmatrix} \quad (2b)$$

$$C = \begin{bmatrix} 1 & -B_m & -TF & 0 & 1 & B_m & 0 & 0 & TF & 0 & 1 & 0 & 0 & 0 \\ 0 & TF & -R & 1 & 0 & 0 & R & 1 & 0 & 1 & 0 & -TF & 0 & 1 \end{bmatrix}^T, \quad (2c)$$

$$D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T. \quad (2d)$$

### Steady-state performance analysis

Let's begin by defining the system parameters.

```
Kt_spec = 13.7; % oz-in/A ... torque constant from spec
Kv_spec = 10.2; % V/krpm ... voltage constant from spec
Tmax_spec = 2.82; % N-m ... max (stall) torque from spec
Omax_spec = 628; % rad/s ... max speed (no load) from spec
N_oz = 0.278013851; % N/oz
m_in = 0.0254; % m/in
Kt_si = Kt_spec*N_oz*m_in; % N-m/A
rads_krpm = 1e3*2*pi/60; % (rad/s)/krpm
Kv_si = Kv_spec/rads_krpm; % V/(rad/s)
Jm = 56.5e-6; % kg-m^2 ... inertia of rotor
Bm = 16.9e-6; % N-m/s^2 ... motor damping coef
R = 1.6; % Ohm ... armature resistance
L = 4.1e-3; % H ... armature inductance
TF = Kv_si; % N-m/A ... trans ratio/motor constant
```

<sup>11</sup>Here is the `rnd` file for use with `statemint.stmartin.edu` to derive the state-space model from the elemental, continuity, and compatibility equations.

Let's investigate what happens in steady-state  $\bar{x}$ . The system is stationary when  $\dot{x} = 0$  and  $u = \bar{u}$  (stationary),<sup>12</sup> so

$$\begin{aligned} 0 &= A\bar{x} + B\bar{u} \Rightarrow \\ \bar{x} &= -A^{-1}B\bar{u}. \end{aligned} \quad (3)$$

Let's compute our steady-state solution for a constant voltage input  $V_s(t) = \bar{V}$  and braking torque  $T_s(t) = \bar{T}$ . We use a symbolic approach to gain insight.

```
syms B_ J_ TF_ L_ R_ Vs_ Ts_ % using underscore for syms

a_ = [-B_/J_, TF_/J_; -TF_/L_, -R_/L_];
b_ = [0, -1/J_; 1/L_, 0];
u_ = [Vs_; Ts_];

M1_ = -inv(a_)*b_ % matrix -A^-1 B
den_ = TF_^2 + B_*R_; % common den
M2_ = M1_.*den_; % factor
xs_ = M1_*u_ % full ss sol
xs_2_ = M2_*u_; % naughty factorless ss sol
```

```
M1_ =

[ TF_/(TF_^2 + B_*R_), -R_/(TF_^2 + B_*R_)]
[ B_/(TF_^2 + B_*R_), TF_/(TF_^2 + B_*R_)]

xs_ =

(TF_*Vs_)/(TF_^2 + B_*R_) - (R_*Ts_)/(TF_^2 + B_*R_)
(B_*Vs_)/(TF_^2 + B_*R_) + (TF_*Ts_)/(TF_^2 + B_*R_)
```

```
eig(a_)
```

<sup>12</sup>A stationary input  $\bar{u}$  is required for a stationary state if the input has any effect on the state; that is, if B is nonzero.

ans =

$$\begin{aligned} & -((B_{-}^2 L_{-}^2 - 2B_{-} J_{-} L_{-} R_{-} + J_{-}^2 R_{-}^2 - 4J_{-} L_{-} T F_{-}^2)^{(1/2)} + B_{-} L_{-} + \\ & \hookrightarrow J_{-} R_{-}) / (2J_{-} L_{-}) \\ & -(B_{-} L_{-} - (B_{-}^2 L_{-}^2 - 2B_{-} J_{-} L_{-} R_{-} + J_{-}^2 R_{-}^2 - 4J_{-} L_{-} T F_{-}^2)^{(1/2)} + \\ & \hookrightarrow J_{-} R_{-}) / (2J_{-} L_{-}) \end{aligned}$$

A little more human-readably, using the fact that  $\Omega_2 = \Omega_J$  and  $i_1 = i_L$ , and using bars to denote steady-state values,

$$\bar{\Omega}_2 = \frac{1}{TF^2 + B_m R} (TF \bar{V}_s - R \bar{T}_s) \quad (4)$$

$$\bar{i}_1 = \frac{1}{TF^2 + B_m R} (B \bar{V}_s + TF \bar{T}_s) \quad (5)$$

Let's focus on the first of these, the relationship between  $\bar{\Omega}_2$  and  $\bar{T}_s$ . For given  $\bar{V}_s$ , there is a linearly decreasing relationship between  $\bar{\Omega}_2$  and  $\bar{T}_s$ . This is precisely the *motor curve*. But it's one of a few curves plotted versus  $\bar{T}_s$ .

Other common curves are current  $\bar{i}_1$ , mechanical braking power

$\mathcal{P}_{\text{brk}} = \bar{T}_s \bar{\Omega}_s$ , and efficiency  $\varepsilon$ . The efficiency is defined as the ratio of the braking power to the voltage source power  $\mathcal{P}_{\text{src}} = \bar{I}_s \bar{V}_s$ ; i.e.

$$\varepsilon = \mathcal{P}_{\text{brk}} / \mathcal{P}_{\text{src}}. \quad (6)$$

We already have expressions for  $\bar{\Omega}_2$  and  $\bar{i}_1$  in terms of  $\bar{T}_s$ , but we must still derive them for  $\mathcal{P}_{\text{brk}}$  and  $\varepsilon$ . For  $\mathcal{P}_{\text{brk}}$ , we must express  $\bar{\Omega}_s$  in terms of known quantities. From the linear graph, it is obvious that  $\bar{\Omega}_s = \bar{\Omega}_2$ . Therefore,

$$\mathcal{P}_{\text{brk}} = \bar{T}_s \bar{\Omega}_2. \quad (7)$$

Now for  $\varepsilon$ . We have the unknown source current  $\bar{I}_s$ . However, from the linear graph, it is obvious that  $\bar{I}_s = \bar{i}_1$ . Therefore,

$$\varepsilon = \frac{\bar{T}_s \bar{\Omega}_2}{\bar{i}_1 \bar{V}_s}. \quad (8)$$

Let's compute these quantities for our parameters.

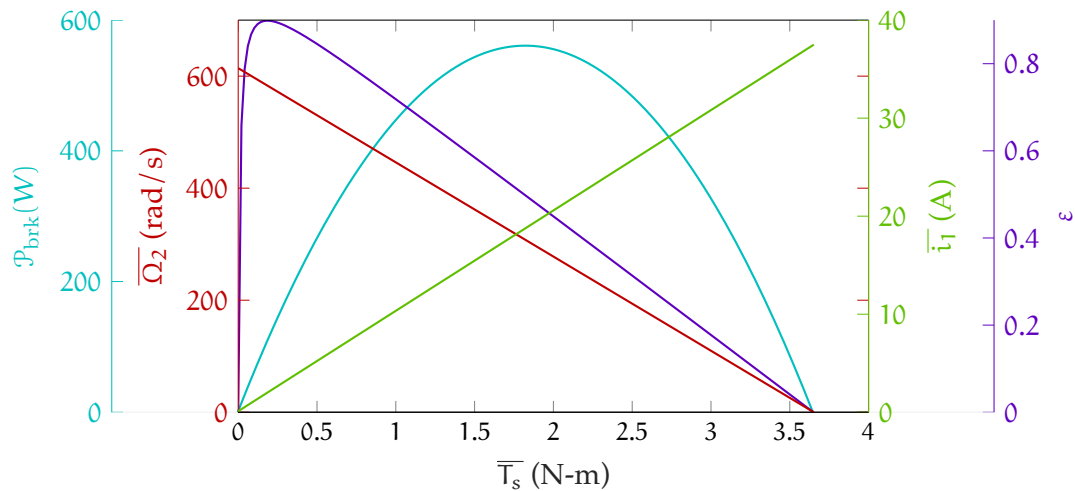


```

Vs = 60; % V ... max used, which is common
Tmax = TF/R*Vs; % N-m ... occurs when Omega_J = 0
Ts_a = linspace(0,Tmax,180); % N-m ... braking torques
O2_a = 1/(TF^2 + Bm*R)*(TF*Vs-R*Ts_a); % rad/s ... ss speed
i1_a = 1/(TF^2 + Bm*R)*(Bm*Vs+TF*Ts_a);
Pbrk_a = Ts_a.*O2_a; % W ... braking power
eff_a = Pbrk_a./(i1_a*Vs);

```

Now let's plot them! The output is shown in Figure curves.2.



**Figure curves.2:** motor curves derived from the model.

There are some key quantities that can be read from the graph and found analytically. The most important are the maximum speed  $\bar{\Omega}_{2_{\max}}$ , which occurs at zero torque, and maximum torque  $\bar{T}_{s_{\max}}$ , which occurs at zero speed. Another is that the maximum mechanical power (output) occurs at  $\bar{T}_{s_{\max}}/2$ . Finally, the maximum efficiency occurs at relatively low torque and high speed, which is typical for the following reason: the two energy-dissipative elements, the resistor and the damper, trade-off as being the dominant effect at the peak, and the resistor tends to dominate. That is, at high speed/voltage and low torque/current, the damper dominates dissipation; at low speed/voltage and high torque/current, the resistor

dominates dissipation. It is very common for a motor's resistance to dominate the damping, as in our case.

Let's examine the maximum speed and torque.

```
Omax = O2_a(1) % rad/s ... occurs when T_s = 0
Tmax % N-m ... already computed and occurs when Omega_2 = 0
```

```
Omax =
    614.2479

Tmax =
    3.6526
```

Comparing these to the values given in the spec sheet, we see we're pretty good, but there's a bit of a discrepancy in the max torque.

```
Omax_spec
Tmax_spec
disp(sprintf('percent error for speed: %0.3g',...
    (Omax-Omax_spec)/Omax_spec*100))
disp(sprintf('percent error for torque: %0.3g',...
    (Tmax-Tmax_spec)/Tmax_spec*100))
```

```
Omax_spec =
    628

Tmax_spec =
    2.8200

percent error for speed: -2.19
percent error for torque: 29.5
```

We should investigate further, but what we will find is that these values are fairly sensitive to TF, B, and R. In our case, it is likely that the given value for

R is a bit low. It is given as  $1.6 \Omega$ , but it is probably closer to  $2 \Omega$ . However, the datasheet for this motor was not clear about whether the maximum speed and torque values were derived from a full motor curve fit or if they were the only points measured. The former is best for estimating dynamic model parameters like R and B, but the latter is occasionally sufficient.

## 04.6 emech.dcmtrans Transient DC motor performance

Let's begin by defining the system parameters.

```
Kt_spec = 13.7; % oz-in/A ... torque constant from spec
Kv_spec = 10.2; % V/krpm ... voltage constant from spec
Tmax_spec = 2.82; % N-m ... max (stall) torque from spec
Omax_spec = 628; % rad/s ... max speed (no load) from spec
N_oz = 0.278013851; % N/oz
m_in = 0.0254; % m/in
Kt_si = Kt_spec*N_oz*m_in; % N-m/A
rads_krpm = 1e3*2*pi/60; % (rad/s)/krpm
Kv_si = Kv_spec/rads_krpm; % V/(rad/s)
d = 2.5*m_in; % m ... flywheel diameter
thick = 1*m_in; % m ... flywheel thickness
vol = pi*(d/2)^2*thick; % flywheel volume
rho = 8000; % kg/m^3 ... flywheel density (304 stainless)
m = rho*vol; % kg ... flywheel mass
Jf = 1/2*m*(d/2)^2; % kg-m^2 ... inertia of flywheel
Jr = 56.5e-6; % kg-m^2 ... inertia of rotor
J = Jf+Jr; % kg-m^2 ... total inertia
Bm = 16.9e-6; % N-m/s^2 ... motor damping coef
Bd = 20e-6; % N-m/s^2 ... bearing damping coef
B = Bm + Bd; % N-m/s^2 ... total damping coef
R = 1.6; % Ohm ... armature resistance
L = 4.1e-3; % H ... armature inductance
TF = Kv_si; % N-m/A ... trans ratio/motor constant
```

The state-space model was derived in [Lecture 04.4 emech.real](#). First, we construct the A, B, C, and D matrices (a, b, c, and d). Then we define a *MATLAB* LTI system model using the `ss` command.

```
a = [-B/J, TF/J; -TF/L, -R/L];
b = [0; 1/L];
c = [1, 0; -B, TF; -TF, -R; 0, 1; 1, 0; B, 0; ...
     0, R; 0, 1; TF, 0; 0, 1; 1, 0; 0, -TF; 0, 0; 0, 1];
```

```
d = [0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0];
sys = ss(a,b,c,d);
```

### Simulating the step response

The step input is widely used to characterize the transient response of a system. *MATLAB*'s `step` function conveniently simulates the step response of an LTI system model.

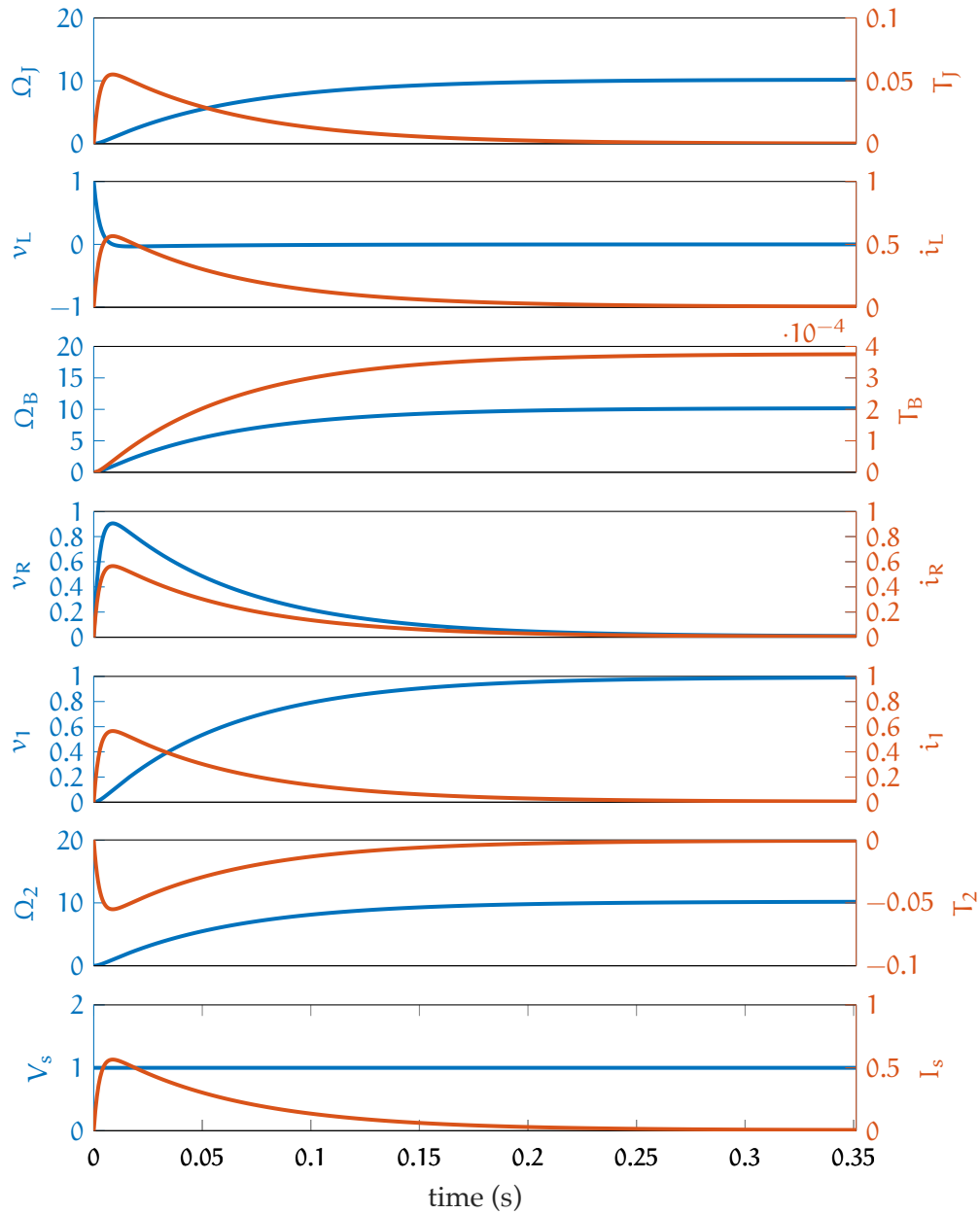
```
[ys_a,t_a] = step(sys);
disp([t_a(1:6),ys_a(1:6,1:4)]) % print a little
```

0	0	0	1.0000	0
0.0002	0.0018	0.0056	0.9082	0.0573
0.0005	0.0071	0.0106	0.8245	0.1093
0.0007	0.0155	0.0152	0.7482	0.1565
0.0010	0.0267	0.0194	0.6786	0.1993
0.0012	0.0405	0.0232	0.6151	0.2381

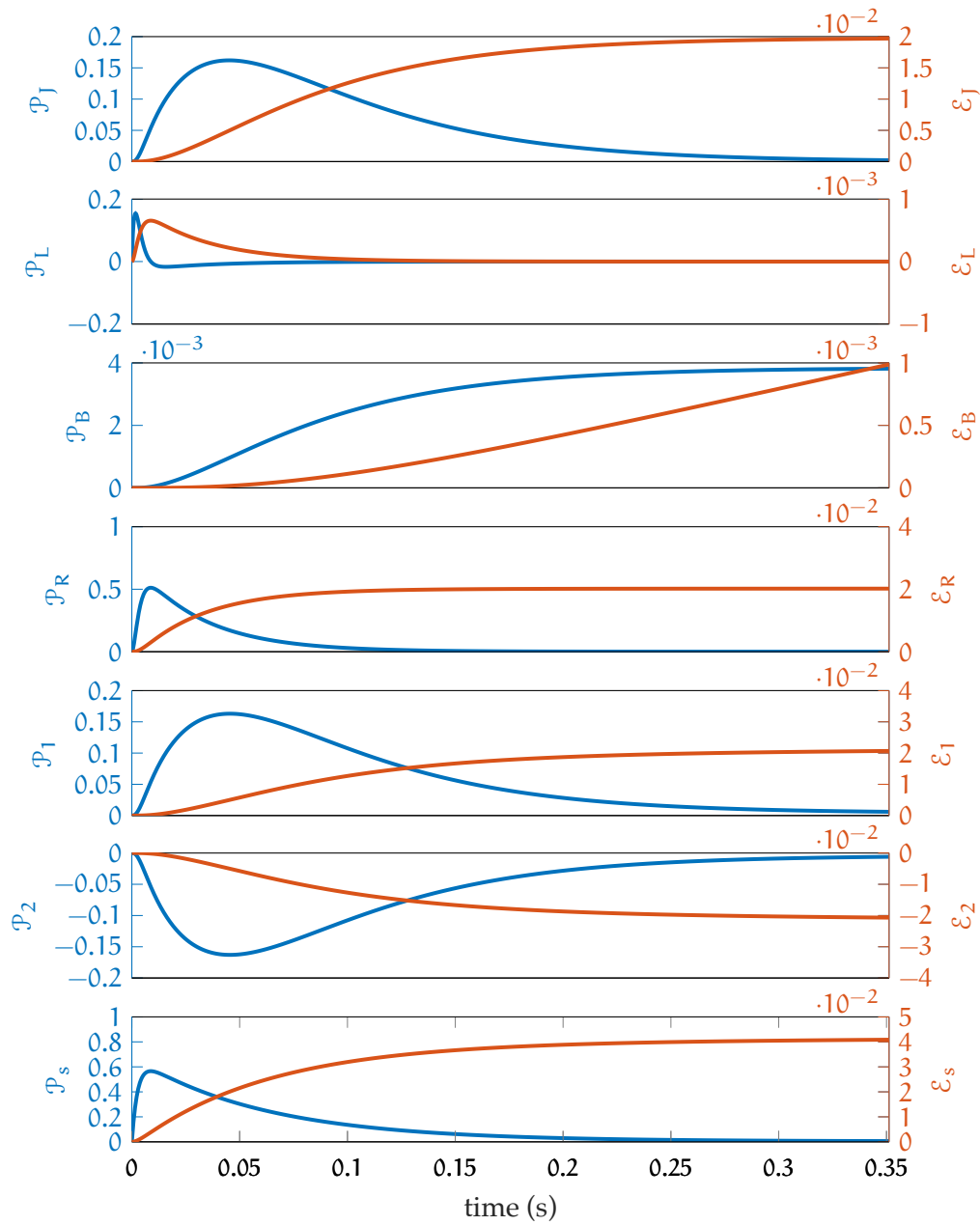
The vector `t_a` contains values of time and array `ys_a` contains a vector of time-series values for each output. If one would like the output for a step input  $ku_s(t)$  (scaled *unit* step  $u_s(t)$ ), by the principle of superposition for linear systems, one can scale the output by  $k$ . The outputs are plotted in [Figure dcmtrans.1](#).

It is also interesting to inspect the power flow and energy associated with each element. Since we have simulated both the across and the through variable for each element, we can compute the instantaneous power by simply taking the product of them at each time step. Moreover, we can cumulatively compute the energy contribution of that power for each element. For energy storage elements, this is the change in energy stored or supplied; for energy dissipative elements, this is the change in energy dissipated; for source elements, this is the energy supplied or absorbed. The results are plotted in [Figure dcmtrans.2](#).

```
P = NaN*ones(size(ys_a,1),size(ys_a,2)/2);
E = NaN*ones(size(P));
```



**Figure dcmtrans.1:** unit step responses for across- (left axes) and through-variables (right axes). Units are as follows: voltage is in V, current is in A, angular velocity is in rad/s, and torque is in N-m. and.



**Figure dcmtrans.2:** power flow (left axes) and energy storage/dissipation/transformation (right axes) for a unit step response. The unit of power is W and the unit of energy is J.

```

j = 0;
for i = 1:2:size(ys_a,2)
    j = j+1;
    P(:,j) = ys_a(:,i).*ys_a(:,i+1);
    E(:,j) = cumtrapz(t_a,P(:,j));
end
disp('power:');
disp(P(1:6,1:4)) % print a little
disp('energy change:');
disp(E(1:6,1:4)) % print a little

```

```

power:
      0      0      0      0
0.0000  0.0520  0.0000  0.0052
0.0001  0.0901  0.0000  0.0191
0.0002  0.1171  0.0000  0.0392
0.0005  0.1352  0.0000  0.0635
0.0009  0.1465  0.0000  0.0907
energy change:
1.0e-03 *
      0      0      0      0
0.0000  0.0064  0.0000  0.0006
0.0000  0.0239  0.0000  0.0036
0.0001  0.0494  0.0000  0.0108
0.0001  0.0805  0.0000  0.0235
0.0003  0.1152  0.0000  0.0425

```

### Estimating parameters from the step response

Often, our model has a couple parameters we don't know well from the specifications, but must attempt to measure. For the system under consideration, perhaps the two parameters most interesting to measure are the dominant time constant and the transformer ratio TF (most important). In this section, we explore how one might estimate them from a measured step response. Other parameters in the system could be similarly estimated. By way of the transfer function, the state-space model can be transformed into input-output differential equations.

```

syms B_ J_ TF_ L_ R_ Vs_ s % using underscore for syms

```



```
a_ = [-B_/J_, TF_/J_ ; -TF_/L_, -R_/L_];
b_ = [0; 1/L_];

(s*eye(2)-a_)^-1*b_
```

```
ans =
      TF_/(TF_^2 + B_*R_ + B_*L_*s + J_*R_*s + J_*L_*s^2)
(B_ + J_*s)/(TF_^2 + B_*R_ + B_*L_*s + J_*R_*s + J_*L_*s^2)
```

The differential equation for  $\Omega_J$  is

$$\frac{d^2\Omega_J}{dt^2} + \left(\frac{R}{L} + \frac{B}{J}\right) \frac{d\Omega_J}{dt} + \frac{TF^2 + BR}{JL} \Omega_J = \frac{TF}{JL} V_s. \quad (1)$$

The corresponding characteristic equation is

$$\lambda^2 + \left(\frac{R}{L} + \frac{B}{J}\right) \lambda + \frac{TF^2 + BR}{JL} = 0 \quad (2)$$

which has solution

$$\lambda_{1,2} = -\frac{1}{2} \left(\frac{R}{L} + \frac{B}{J}\right) \pm \frac{1}{2} \sqrt{\left(\frac{R}{L} + \frac{B}{J}\right)^2 - 4 \frac{TF^2 + BR}{JL}}. \quad (3)$$

For a step input  $V_s(t) = \bar{V}_s$ ,  $\Omega_J(0) = d\Omega_J(0)/dt = 0$ , and distinct roots  $\lambda_1$  and  $\lambda_2$ , the solution is

$$\Omega_J(t) = \bar{V}_s \frac{TF}{TF^2 + BR} \left(1 - \frac{1}{\lambda_2 - \lambda_1} (\lambda_2 e^{\lambda_1 t} - \lambda_1 e^{\lambda_2 t})\right) \quad (4)$$

Let's compute  $\lambda_1$  and  $\lambda_2$ .

```
lambda12 = -1/2*(R/L+B/J) + ...
          [1,-1]*1/2*sqrt((R/L+B/J)^2 - 4*(TF^2+B*R)/(J*L))
```

```
lambda12 =
-16.3467 -373.9941
```

Both values are *real*, so we expect not an oscillation, but a decay to a final value. However, that decay occurs with two different time constants:

$$\tau_1 = -1/\lambda_1 \text{ and } \tau_2 = -1/\lambda_2.$$

```
tau12 = -1./lambda12
disp(['ratio: ', num2str(tau12(1)/tau12(2))])
```

```
tau12 =
    0.0612    0.0027
ratio: 22.8788
```

So second decays much faster than the first. That's good news for our estimation project because we can easily ignore the step response's first  $5\tau_2 \approx 0.0134$  s and assume the rest is decaying at  $\tau_1$ , which we call the *dominant time constant* and which we would like to estimate.

Let's generate some fake response data to get the idea. We'll layer on some Gaussian noise with `randn` to be more realistic. The data is plotted in [Figure dcmtrans.3](#).

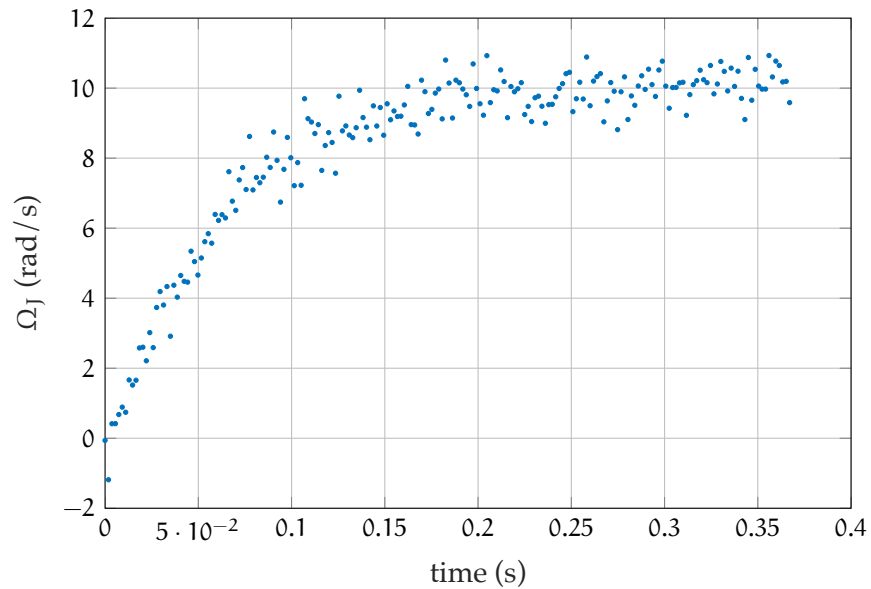
```
t_data = linspace(0, -6/lambda12(1), 200);

O_fun = @(t) TF/(TF^2+B*R)*...
    (1-1/(lambda12(2)-lambda12(1))*...
    (lambda12(2)*exp(lambda12(1)*t)-...
    lambda12(1)*exp(lambda12(2)*t)));
rng(2);
O_data = O_fun(t_data) + .5*randn(size(t_data));
```

Let's trim the data to eliminate the time interval corresponding to the first five of the "fast" time constant  $\tau_2$ .

```
[t_5, i_5] = min(abs(t_data - (-5/lambda12(2)))); % delete
t_data_trunc = t_data((i_5+1):end);
O_data_trunc = O_data((i_5+1):end);
```

We need want to take the natural logarithm of the data so we can perform a linear regression to estimate the "experimental" slow time constant  $\tilde{\tau}_1$ . We must first estimate the steady-state value  $\Omega_{J\infty}$  (which we'll also need). We don't want to just take the last value in the array due to its noisiness. The data goes for six slow time constants, so averaging the data for the last time constant is a good estimate.



**Figure dcmtrans.3:** unit step response "data."

```
[t_ss,i_ss] = ...
    min(abs(t_data_trunc-(-5/lambda12(1)))); % start here
O_data_ss = O_data_trunc((i_ss+1):end);
mu_O_ss = mean(O_data_ss)
S_mu_O_ss = std(O_data_ss)/sqrt(length(O_data_ss))
```

```
mu_O_ss =
    10.1801
S_mu_O_ss =
    0.0763
```

Let's use this result to transform the data into its linear form.

```
O_lin = log(-(O_data_trunc-mu_O_ss));
O_lin_complex = find(imag(O_lin)>0);
disp(['number of complex values: ',...
    num2str(length(O_lin_complex))])
```

```
number of complex values: 33
```

Now we have encountered a problem. The noisiness of the data makes some of our points wander into negative-land. Logarithms of negative numbers are complex. Naive approaches like just taking real parts, excluding complex values, or coercing complex values to  $-\infty$  all have the issue of biasing the data.

There are a lot of approaches we could take. The best approaches include nonlinear regression and discrete filtering to smooth the data (e.g. `filtfilt`).

We opt for an easier approach: we find the index at which the time series first transgresses the boundary and exclude the data beyond the previous index.

```
i_bad = 0_lin_complex(1);
t_lin_trunc = t_data_trunc(1:i_bad-1);
0_lin_trunc = 0_lin(1:i_bad-1);
```

This is plotted in [Figure dcmtrans.4](#) along with the linear regression least-squares fit, computed below.

```
pf = polyfit(t_lin_trunc,0_lin_trunc,1);
0_lin_fit = polyval(pf,t_lin_trunc);
tau_1_est = -1/pf(1)
```

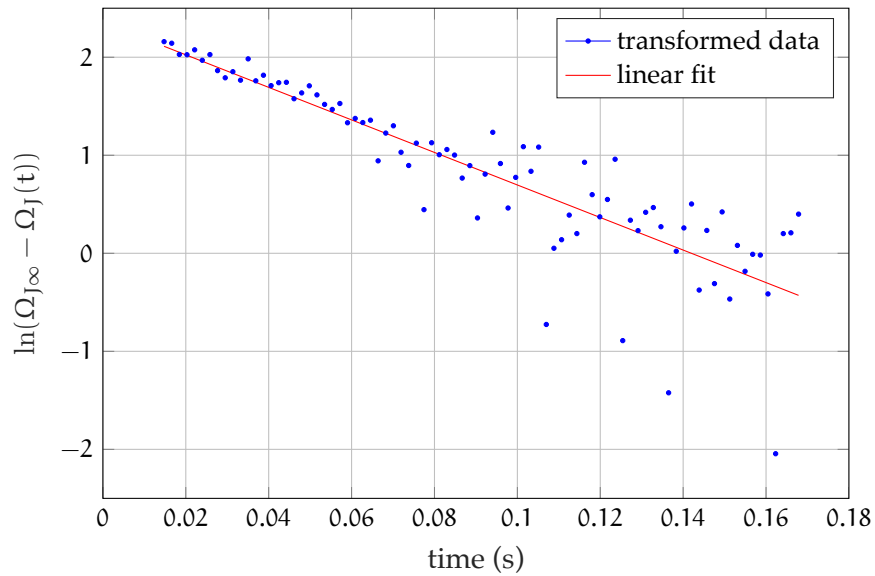
```
tau_1_est =
    0.0603
```

So our estimate for  $\tau_1$  is  $\tilde{\tau}_1 = 60.3$  ms. Recall that our analytic expression for  $\tau_1$  is known in terms of other parameters. Similarly, the steady-state value of  $\Omega_J$ , which has already been estimated to be  $\Omega_{J\infty} = 10.18$  (i.e. `mu_0_ss`). This occurs when the time-derivatives of  $\Omega_J$  are zero. From the solution for  $\Omega_J$  (or its differential equation), for constant  $V_s(t) = \bar{V}_s$ , this occurs when

$$\Omega_{J\infty} = \frac{TF}{TF^2 + BR} \bar{V}_s. \quad (5)$$

An analytic expression for TF can be found by solving [Equation 5](#), which yields

$$TF = \bar{V}_s \pm \frac{1}{2\tilde{\Omega}_{J\infty}} \sqrt{V_s^2 - 4BR\tilde{\Omega}_{J\infty}^2} \quad (6)$$



**Figure dcmtrans.4:** transformed angular velocity “data” with a linear fit.

We choose the solution closer to the *a priori* (spec) value of 0.0974.

$$TF\_est = (1 + (-4*B*R*\mu_{0\_ss}^2 + 1^2)^{1/2}) / (2*\mu_{0\_ss})$$

$$TF\_est = 0.0976$$

This estimate  $\tilde{T}F = 0.0976$  is very close to the value given in the specification sheet *because we constructed it to be so*. Real measurements would probably yield an estimate further from the specification, which is why we would estimate it.

## 04.7 emech.drive Driving motors

1 The DC motor requires DC electrical power provided by a circuit called the “driving” circuit. For industrial motors at least, these circuits must provide significant power, and for this reason a separate (from the control circuit) power supply is often used. There is a quick-and-dirty way to drive a DC motor at variable speed: since its angular velocity is reliably proportional to its voltage, place a potentiometer in series with the power supply and motor. However, this has disadvantages that include the power being limited and dissipated at high potentiometer resistance (low speed). For most applications, we will need either a current (or power) amplifier or—more likely—a microcontroller and an integrated circuit to produce a pulse-width modulation driving signal.

### *Pulse-width modulation*

2 *Pulse-width modulation* (PWM) is a technique used to deliver an effectively variable signal to a load (in this case a motor) without a truly variable power source. A pulse of full source amplitude is repeated at a high frequency (e.g. 20 kHz), delivering a signal that is effectively averaged by the load dynamics such that its effects on the load are nearly continuous. The fraction of the period that the signal is high (on) is called the *duty cycle*  $\delta$ . The following figure shows a PWM signal  $v(t)$  and its average  $\bar{v}(t)$  with a few parameter definitions.

3 The mean of any periodic signal can be computed with the integral

$$\bar{v}(t) = \frac{1}{T} \int_0^T v(t),$$

which is easily evaluated for a PWM signal:

$$\bar{v}(t) = \frac{Aw}{T} = A\delta.$$

4 This result shows that if a PWM signal is delivered to a load, such as a DC motor, that is relatively unaffected by high-frequency signals, the effective signal will be simply the product of the source amplitude  $A$  and the

duty cycle  $\delta$ . The duty cycle can have values from 0 to 1, so the effective DC signal produced varies linearly  $\delta$  from 0 to  $A$ .

*PWM with a microcontroller and integrated circuit*

**5** A microcontroller such as the myRIO or Arduino can easily produce a PWM signal; however, this signal is typically *low-power* and cannot drive even small DC motors. Therefore it is common to include a special kind of integrated circuit (IC) that uses the microcontroller's low-power PWM signal to gate a high-power DC source signal for delivery to the motor. We use a connectorized printed circuit board (PCB, e.g. a PC motherboard)—the [Pololu motor driver carrier](#)—that includes on it a [STMicroelectronics VN1500](#) H-bridge motor driver *integrated circuit* (IC, i.e. a microchip).

**H-bridge circuits** **6** We want to drive DC motors at different effective voltages *and* different directions. An H-bridge circuit allows us to reverse the direction of the PWM signal delivered to the motor. The following is a diagram of the H-bridge circuit.

**7** The switches S1-S4 are typically instantiated with MOSFET transistors. As shown in the figure below, during the high duration of the PWM pulse, either S1 and S4 (a) or S2 and S3 (b) are closed and the others are open.

- (a) motor driven one direction
- (b) motor driven the opposite direction

**8** Recall that a DC motor can be modeled as a resistor and inductor in series with an electro-mechanical transformer. The inductance of the windings make it an “inductive” load, which presents the following challenge. We can't rapidly change the current flow through an inductor without a huge spike in voltage, and the switches do just that, leading to switch damage. Therefore, during the low or “off” duration of the PWM signal, S1-S4 cannot all be simply opened. There are actually a few options for switch positions that allow the current to continue to flow without inductive “kickback.”

**9** What's up with the diodes? Technically, they could be used to deal with the kickback, but they typically are not because they dissipate power.

However, they are used to do just that to ease the transition between switch flips, which are never quite simultaneous.

### Motor curves

**10** Motors are often characterized by three *steady-state* curves:

1. a torque  $T$  versus angular velocity  $\Omega$  curve;
2. an angular velocity  $\Omega$  versus voltage  $v$  curve, which has slope  $1/k_m$ ;  
and
3. a torque  $T$  versus current  $i$  curve, which has slope  $-k_m$ .

**11** We will develop our own motor curves for the DC motor in the lab by simultaneously measuring  $v$ ,  $i$ , and  $\Omega$ . Unfortunately, we will not be measuring  $T$  directly, and so we will be unable to measure all these curves directly; however, we will be able to infer them based on the (reasonable, but not perfect) assumption that the motor has no power losses. In the end, they should look something like the following (using our usual sign convention).

**12** In order to construct such curves, we will measure  $v$ ,  $i$ , and  $\Omega$ . The following sections describe the measurement process.



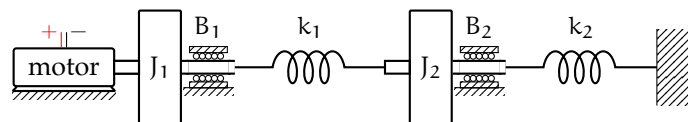
## 04.8 emech.exe Exercises for Chapter 04 emech

### Exercise 04.1 triangle

Respond to the following questions and imperatives with one or two sentences and, if needed, equations and/or sketch.

- Why do we include a resistor in lumped-parameter motor models?
- How are brushes used in brushed DC motors?
- With regard to standard motor curves, why do we say the “braking power” is equivalent to the power that could be successfully transferred by the motor to the mechanical system?
- In terms of electrical and mechanical processes, why does an *efficiency* versus torque motor curve have a peak?
- As a DC motor’s bearings wear down, how will its efficiency curve be affected?

### Exercise 04.2 square



**Figure exe.1:** schematic of an electromechanical system for Exercises 04.2 emech. and 04.3 emech..

Consider the system presented in the schematic of Fig. exe.1. Let the DC motor have motor constant  $K_a$  (units N-m/A) and let the motor be driven by an ideal *current* source  $I_S$ . Assume the motor inertia has been lumped into  $J_1$  and motor damping lumped into  $B_1$ .

- Draw a linear graph model.
- Draw a normal tree.

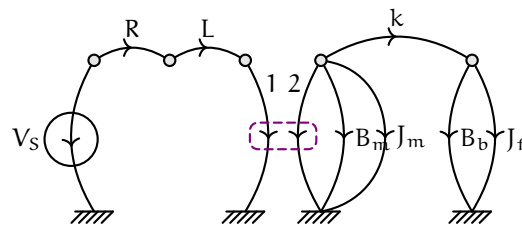
- c. Identify any *dependent* energy storage elements. If the motor was driven by an ideal voltage source instead, how would this change?

Draw a linear graph model and normal tree.

### Exercise 04.3 rectangle

Consider the system presented in the schematic of Fig. exe.1. From the linear graph model and normal tree derived in Exercise 04.2 emech., derive a state-space model in standard form. Let the outputs be  $\theta_{J_1}$  and  $\theta_{J_2}$ , the angular positions of the flywheels.

### Exercise 04.4 quadrilateral



**Figure exe.2:** a linear graph model of the electromechanical system.

Consider the linear graph model of a motor coupled to a rotational mechanical system shown in Fig. exe.2. This is similar to the model from the Lec. 04.4 emech.real, but includes the flexibility of the shaft coupler. An ideal voltage source drives the motor—modeled as an ideal transducer with armature resistance  $R$  and inductance  $L$ , given by the manufacturer in Table real.1. The ideal transducer's rotational mechanical side (2) is connected to a moment of inertia  $J_m$  modeling the rotor inertia and damping  $B_b$  modeling the internal motor damping, both values given in the motor specifications. Take  $B_b = B_m$  and  $J_f = 0.324 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$ . Assume the shaft coupling has a torsional stiffness of  $k = 100 \text{ N}\cdot\text{m}/\text{rad}$ .

- a. Derive a state-space model for the system with outputs  $i_1$  and  $\Omega_{J_f}$ .

- b. Create a Matlab ss model for the system and simulate its response from rest to an input voltage  $V_S = 10$  V.
- c. Plot the outputs through time until they reach steady state.

### Exercise 04.5 mrpotatohead

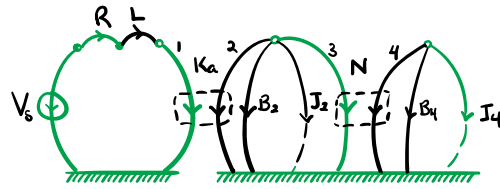
Consider the linear graph model (with normal tree) of Fig. exe.3. This is a model of a motor with constant  $K_a$  connected to a pair of meshing gears with transformer ratio  $N$ , the output over input gear ratio. An ideal voltage source drives the motor—modeled as an ideal transducer with armature resistance  $R$  and inductance  $L$ . The motor's rotational mechanical side (2) is connected to a moment of inertia  $J_2$  modeling the rotor and drive gear combined inertia. The damping element  $B_2$  models the internal motor damping and the drive gear bearing damping. The output side of the gear transducer (4) is connected to a moment of inertia  $J_4$  modeling the output gear and load combined inertia. The damping element  $B_4$  models the internal motor damping and the drive gear bearing damping. Use the parameter values given in Table exe.1.

\_\_\_\_\_/  
30 p.

- a. Derive a state-space model for the system with outputs  $\Omega_{J_2}$  and  $\Omega_{J_4}$ .
- b. Create a Matlab ss model for the system and simulate its response from rest to an input voltage  $V_S = 20$  V.
- c. Plot the outputs through time until they reach steady state.

**Table exe.1: System parameter values.**

R	2 $\Omega$
L	8 mH
$K_a$	0.2 N-m/A
$J_2$	$0.1 \cdot 10^{-3}$ kg-m <sup>2</sup>
$B_2$	50 $\mu$ N-m/(rad/s)
N	5
$J_4$	$1 \cdot 10^{-3}$ kg-m <sup>2</sup>
$B_4$	70 $\mu$ N-m/(rad/s)



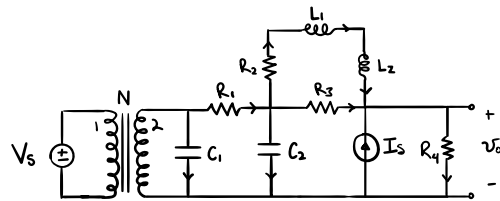
**Figure exe.3:** A linear graph model with normal tree in green of an electromechanical system with a gear reduction.

**Exercise 04.6 clunker**

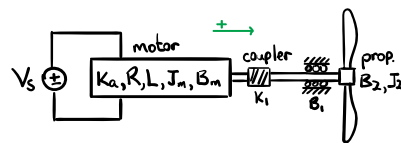
Draw a *linear graph*, a *normal tree*, identify *state variables*, identify *system order*, and denote any *dependent energy storage elements* for each of the following schematics.

\_\_\_\_\_/ 25 p.

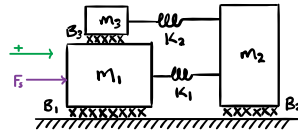
- a. The electronic system of Fig. exe.4, voltage and current sources, and transformer with transformer ratio  $N$ .
- b. The electromechanical system of Fig. exe.5 with motor model parameters shown, coordinate arrow in green. Model the propeller as a moment of inertia  $J_2$  and damping  $B_2$ .
- c. The translational mechanical system of Fig. exe.6, force source, coordinate arrow in green.



**Figure exe.4:** a circuit diagram.



**Figure exe.5:** Sketch of a motor coupled to a fan.



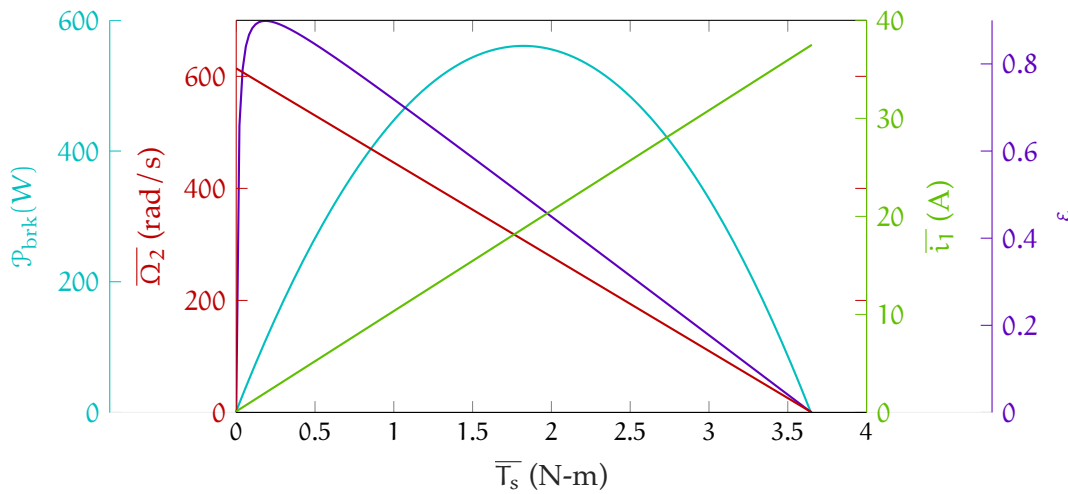
**Figure exe.6:** Schematic of a mechanical system.

**Exercise 04.7 curvy**

Consider the DC motor curves of Fig. curves.2, reproduced in Fig. exe.7.

\_\_\_\_\_/ 20 p.

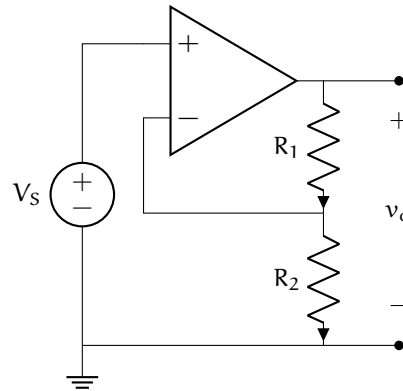
- a. At peak efficiency, what is the steady-state motor speed?
- b. At peak efficiency, what is the steady-state motor torque?
- c. You are to use this motor to drive a load at a constant angular speed of 100 rad/s with at least 1 N-m of torque. You wisely choose to use a gear reduction between the motor and load. What should the gear ratio be to meet the above requirements and optimize efficiency? Justify your answer in terms of the motor curves of Fig. exe.7.



**Figure exe.7:** the motor curve Fig. curves.2.

## Exercise 04.8 chair

---

 25 p.


**Figure exe.8:** An opamp circuit.

Consider the opamp circuit of Fig. exe.8, which will be used to drive a PMDC motor. The input can supply a variable  $V_S \in [0, 10]$  V, the motor has constant  $K_a = 0.05$  V/(rad/s) and coil resistance  $R_m = 1$   $\Omega$ , and the opamp has differential supplies  $\pm 24$  V. Assume the maximum torque magnitude required from the motor at top speed is  $|T_2| = 0.1$  N-m and ignore any voltage drop in the motor due to the coil inductance.<sup>13</sup> Select  $R_1$  and  $R_2$  to demonstrably meet the following *design requirements*:

- a. drivable motor speeds of at least  $[0, 400]$  rad/s,
- b. no saturation of the opamp (i.e.  $|v_o| < 24$  V), and
- c. a maximum combined power dissipation by  $R_1$  and  $R_2$  less than 300 mW.

*Hint:* start with the elemental equations of the DC motor to determine the necessary amplifier output  $v_o$ , then constrain  $R_1$  and  $R_2$  to meet the gain requirements, and finally further constrain  $R_1$  and  $R_2$  to meet the power dissipation requirement.

<sup>13</sup>Do not ignore the voltage drop across  $R_m$ , though. Note that this amounts to an assumption of steady-state operation at top speed. By requiring a specific  $T_2$ , we are also implicitly ignoring torque losses due to motor bearing damping.

**Exercise 04.9 onomatopoeia**

Consider the DC motor curves of Fig. curves.2, reproduced in Fig. exe.7. If this motor is running at  $400 \frac{\text{rad}}{\text{s}}$ ,

1. How much torque is produced?
2. What is the output power?
3. What is the input power?
4. Why are the input and output power the same or different?

**Exercise 04.10 deglazification**

Explain in your own words what lumped parameter elements should be used when modeling an electric motor and why.

\_\_\_\_\_/   
 10 p.

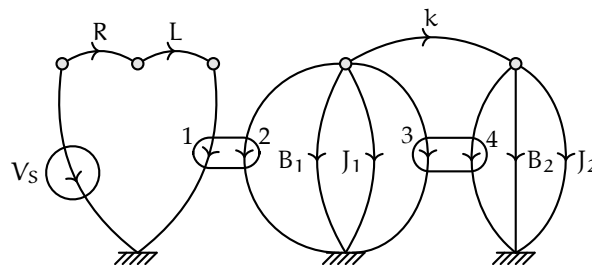
**Exercise 04.11 confuzzled**

In the linear graph below a system is depicted consisting of a motor with its related damping and inertia driven by a voltage source and connected to a set of gears driving a second inertia. A rotary spring is attached between the two inertias.

\_\_\_\_\_/   
 20 p.

Given this linear graph:

1. draw a normal tree,
2. determine the state variables and system order, and
3. list any dependent energy storage elements and explain what this implies.



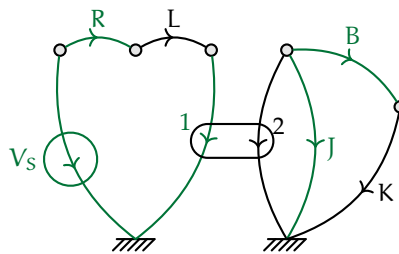
**Exercise 04.12 levitation**

In the linear graph and normal tree below a system is depicted consisting of a motor driven by a voltage source  $V_S$  with inertia  $J$  driving a rotary damper and spring connected in series. Let the motor constant be  $K_a$ , and outputs of the system be the rotational velocity of the inertia,  $\Omega_J$ , and the change in rotational velocity across the rotational damper,  $\Omega_B$ .

\_\_\_\_\_/ 40 p.

Given this linear graph and normal tree:

1. determine the state variables,
2. define the state, input, and output vectors,
3. write the elemental, continuity, and compatibility equations, and
4. solve for the state and output equations.

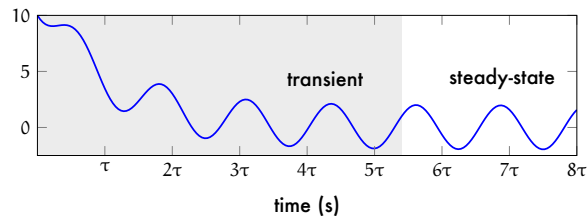




## **Part II**

### **Time response**

- 1 In this chapter, we will extend our understanding of linear, time-invariant (LTI) system properties. We must keep in mind a few important definitions.
- 2 The **transient response** of a system is its response during the initial time-interval during which the initial conditions dominate. The **steady-state response** of a system is its remaining response, which occurs after the transient response. Fig. lti.1 illustrates these definitions.



**Figure lti.1:** transient and steady-state responses. Note that the transition is not precisely defined. (Figure adapted from *Electronics: an introduction* by Picone.)

- 3 The **free response** of a system is the response of the system to initial conditions—*without forcing* (i.e. the specific solution of the io ODE with the forcing function identically zero). This is closely related to, but distinct from, the transient response, which is the free response *plus* an additional term. This additional term is the **forced response**: the response of the system to a forcing function—*without initial conditions* (i.e. the specific solution of the io ODE with the initial conditions identically zero).

## 05.1 lti.super+ Superposition, derivative, and integral properties

1 From the principle of **superposition, linear, time invariant (LTI)** system responses to both initial conditions and nonzero forcing can be obtained by summing the free response  $y_{fr}$  and forced response  $y_{fo}$ :

$$y(t) = y_{fr}(t) + y_{fo}(t).$$

Moreover, superposition says that any linear combination of inputs yields a corresponding linear combination of outputs. That is, we can find the response of a system to each input, separately, then linearly combine (scale and sum) the results according to the original linear combination. That is, for inputs  $u_1$  and  $u_2$  and constants  $a_1, a_2 \in \mathbb{R}$ , a forcing function

would yield output

where  $y_1$  and  $y_2$  are the outputs for inputs  $u_1$  and  $u_2$ , respectively.

2 This powerful principle allows us to construct solutions to complex forcing functions by decomposing the problem. It also allows us to make extensive use of existing solutions to common inputs.

3 There are two more LTI system properties worth noting here. Let a system have input  $u_1$  and corresponding output  $y_1$ . If the system is then given input  $u_2(t) = \dot{u}_1(t)$ , the corresponding output is

Similarly, if the same system is then given input  $u_3(t) = \int_0^t u_1(\tau) d\tau$ , the corresponding output is



These are sometimes called the **derivative** and **integral properties** of LTI systems.

## 05.2 lti.equistab Equilibrium and stability properties

**1** For a system with LTI state-space model  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ ,  $\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$ , the model is in an **equilibrium** state  $\bar{\mathbf{x}}$  if  $\dot{\mathbf{x}} = 0$ . This implies  $\mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\mathbf{u} = 0$ . For constant input  $\bar{\mathbf{u}}$ , this implies

If  $\mathbf{A}$  is invertible,<sup>1</sup> as is often the case, there is a unique solution for a single **equilibrium** state:

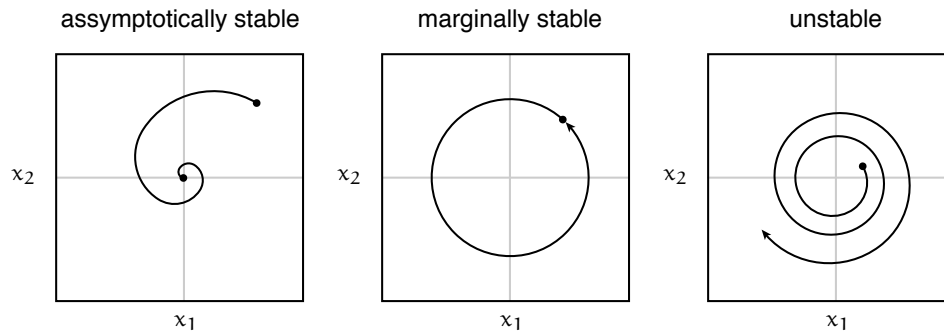
### Definition 05 lti.1: Stability

If  $\mathbf{x}$  is perturbed from an equilibrium state  $\bar{\mathbf{x}}$ , the response  $\mathbf{x}(t)$  can:

1. asymptotically return to  $\bar{\mathbf{x}}$
2. diverge from  $\bar{\mathbf{x}}$
3. remain perturbed or oscillate about  $\bar{\mathbf{x}}$  with a constant amplitude

**2** A **phase portrait** is a parametric plot of state variable **trajectories**, with time implicit. Phase portraits are exceptionally useful for understanding nonlinear systems, but they also give us a nice way to understand stability, as in [Figure equistab.1](#).

<sup>1</sup>If  $\mathbf{A}$  is not invertible, the system has at least one eigenvalue equal to zero, which yields an *equilibrium subspace* equal to an offset (by  $\mathbf{B}\bar{\mathbf{u}}$ ) of the null space of the state space  $\mathbb{R}^n$ .



**Figure equistab.1:** a phase-portrait demonstration of (left) asymptotic stability, (center) marginal stability, and (right) instability for a second-order system.

3 These definitions of stability can be interpreted in terms of the free response of a system, as described, below.

### Stability defined by the free response

4 Using the concept of the free response (no inputs, just initial conditions), we define the following types of stability for LTI systems<sup>2</sup>.

1. An LTI system is **asymptotically stable** if the free response approaches an equilibrium state as time approaches infinity.
2. An LTI system is **unstable** if the free response grows without bound as time approaches infinity.
3. An LTI system is **marginally stable** if the free response neither decays nor grows but remains constant or oscillates as time approaches infinity.

5 These statements imply that the free response alone governs stability. Recall that the free response  $y_{fr}$  of a system with characteristic equation roots  $\lambda_i$  with multiplicity  $m_i$ , for constants  $C_i$ , is



<sup>2</sup> Nise, 2015.

Each term will either decay to zero, remain constant, or increase without bound—depending on the sign of the *real part* of the corresponding root of the characteristic equation  $\text{Re}(\lambda_i)$ .

6 In other words, for an LTI system, the following statements hold.

1. An LTI system is *asymptotically stable* if, for all  $\lambda_i$ ,  $\text{Re}(\lambda_i) < 0$ .
2. An LTI system is *unstable* if, for any  $\lambda_i$ ,  $\text{Re}(\lambda_i) > 0$ .
3. An LTI system is *marginally stable* if,
  - a) for all  $\lambda_i$ ,  $\text{Re}(\lambda_i) \leq 0$  and
  - b) at least one  $\text{Re}(\lambda_i) = 0$  and
  - c) no  $\lambda_i$  for which  $\text{Re}(\lambda_i) = 0$  has multiplicity  $m_i > 1$ .

### Stability defined by the forced response

7 An alternate formulation of the stability definitions above is called the **bounded-input bounded-output** (BIBO) definition of stability, and states the following<sup>3</sup>.

1. A system is **BIBO stable** if every bounded input yields a bounded output.
2. A system is **BIBO unstable** if any bounded input yields an unbounded output.

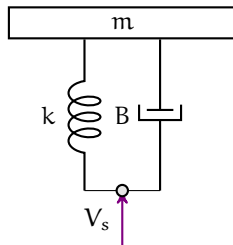
8 In terms of BIBO stability, **marginal stability**, then, means that a system has a bounded response to some inputs and an unbounded response to others. For instance, a second-order undamped system response to a sinusoidal input at the natural frequency is unbounded, whereas every other input yields a bounded output.

9 Although we focus on the definitions of stability in terms of the free response, it is good to understand BIBO stability, as well.

---

<sup>3</sup> Nise, 2015.

## 05.3 Ivi.vib Vibration isolation table analysis

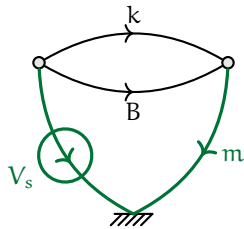


**Figure vib.1:** a vibration isolation table schematic with input velocity  $V_s$ .

- 1 In this example, we exercise many of the methods for modeling and analysis explored thus far.
- 2 Given the vibration isolation table model in [Figure vib.1](#)—with  $m = 320$  kg,  $k = 16000$  N/m, and  $B = 1200$  N-m/s—derive:
  1. a linear graph model,
  2. a state-space model,
  3. the equilibrium state  $\bar{x}$  for the unit step input,
  4. a transfer function model,
  5. an input-output differential equation model with input  $V_s$  and output  $v_m$ ,
  6. a solution for  $v_m(t)$  for a unit step input  $V_s(t) = 1$  m/s for  $t \geq 0$ ,
  7. the system's stability.



Linear graph and state-space models



**Figure vib.2:** linear graph of the isolation table.

3 The linear graph and normal tree are shown in **Figure vib.2**. Note that there is an equilibrium for this system, so we are justified in ignoring gravity and referencing any displacements to the static equilibrium position.<sup>4</sup> The state variables are the velocity of the mass  $v_m$  and the force through the spring  $f_k$  and the system order is  $n = 2$ . The input, state, and output vectors are

$$\mathbf{u} = [V_s] \quad \mathbf{x} = \begin{bmatrix} v_m \\ f_k \end{bmatrix} \quad \mathbf{y} = [v_m].$$

The elemental equations are as follows.

$$\begin{array}{l|l} m & \dot{v}_m = \frac{1}{m} f_m \\ k & \dot{f}_k = kv_k \\ B & f_B = Bv_B \end{array}$$

The continuity and compatibility equations are as follows.

branch	continuity equation	link	compatibility equation
m	$f_m = f_k + f_B$	k	$v_k = V_s - v_m$
		B	$v_B = V_s - v_m$

The state equation can be found by substituting the continuity and compatibility equations into the elemental equations, and eliminating  $f_B$ , to

<sup>4</sup>For a discussion of this ignoring of gravity, see [Lec. 05.4 lti.ghost](#).

yield

$$\dot{\mathbf{x}} = \begin{bmatrix} -B/m & 1/m \\ -k & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} B/m \\ k \end{bmatrix} \mathbf{u} \quad (1a)$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \end{bmatrix} \mathbf{u}. \quad (1b)$$

## Equilibrium

4 Let's check to see if  $A$  is invertible by trying to compute its inverse:

$$A^{-1} = \begin{bmatrix} -B/m & 1/m \\ -k & 0 \end{bmatrix}^{-1} \quad (2)$$

$$= \frac{1}{k/m} \begin{bmatrix} 0 & -1/m \\ k & -B/m \end{bmatrix} \quad (3)$$

So it has an inverse, after all! Let's use this to compute the equilibrium state:

$$\bar{\mathbf{x}} = -A^{-1} B \bar{\mathbf{u}} \quad (4)$$

$$= \frac{-m}{k} \begin{bmatrix} 0 & -1/m \\ k & -B/m \end{bmatrix} \begin{bmatrix} B/m \\ k \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} \quad (5)$$

$$= \frac{-m}{k} \begin{bmatrix} -k/m \\ 0 \end{bmatrix} \quad (6)$$

$$= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (7)$$

So the system is in equilibrium with  $v_m = 1$  m/s and  $f_k = 0$  N. Since  $v_m$  is also our output, we expect 1 m/s to be our steady-state output value.

## Transfer function model

5 The transfer function  $H(s) = V_m(s)/V_s(s)$  will be used as a bridge to the input-output differential equation. The transfer function can be found from the usual formula, from [Lecture 03.7 ss.ss2tf2io](#),

$$H(s) = C(sI - A)^{-1} B + D. \quad (8)$$

Let's first compute  $(s\mathbf{I} - \mathbf{A})^{-1}$ .<sup>5</sup>

$$(s\mathbf{I} - \mathbf{A})^{-1} = \left( \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} -B/m & 1/m \\ -k & 0 \end{bmatrix} \right)^{-1} \quad (9a)$$

$$= \begin{bmatrix} s + B/m & -1/m \\ k & s \end{bmatrix}^{-1} \quad (9b)$$

$$= \frac{1}{(s + B/m)(s) - (-1/m)(k)} \begin{bmatrix} s & 1/m \\ -k & s + B/m \end{bmatrix} \quad (9c)$$

$$= \frac{1}{s^2 + (B/m)s + k/m} \begin{bmatrix} s & 1/m \\ -k & s + B/m \end{bmatrix} \quad (9d)$$

Now we're ready to compute the entirety of  $H$ :

$$H(s) = \frac{1}{s^2 + (B/m)s + k/m} \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s & 1/m \\ -k & s + B/m \end{bmatrix} \begin{bmatrix} B/m \\ k \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \quad (10a)$$

$$= \frac{1}{s^2 + (B/m)s + k/m} \begin{bmatrix} s & 1/m \end{bmatrix} \begin{bmatrix} B/m \\ k \end{bmatrix} \quad (10b)$$

$$= \frac{(B/m)s + k/m}{s^2 + (B/m)s + k/m}. \quad (10c)$$

### Input-output differential equation

**6** The input-output differential equation can be found from the reverse of the procedure in [Lecture 03.7 ss.ss2tf2io](#). Beginning from the transfer function,

$$\frac{V_m}{V_s} = \frac{(B/m)s + k/m}{s^2 + (B/m)s + k/m} \Rightarrow \quad (11a)$$

$$(s^2 + (B/m)s + k/m) V_m = ((B/m)s + k/m) V_s \Rightarrow \quad (11b)$$

$$\ddot{v}_m + (B/m)\dot{v}_m + (k/m)v_m = (B/m)\dot{V}_s + (k/m)V_s. \quad (11c)$$

### Step response

**7** The step response is found using superposition and the derivative property of LTI systems. The forcing function  $f(t) = (B/m)\dot{V}_s + (k/m)V_s$  is

<sup>5</sup>See (Rowell and Wormley, 1997, Sec. A.4.3) for details on the matrix inverse.

composed of two terms, one of which has a derivative of the input  $V_s$ . Rather than attempting to solve the entire problem at once, we choose to find the response for a forcing function  $f(t) = 1$  (for  $t \geq 0$ )—that is, the unit step response—and use superposition and the derivative property of LTI systems to calculate the composite response.

### Unit step response

**8** The characteristic equation of Equation 11c is

$$\lambda^2 + (B/m)\lambda + k/m = 0 \Rightarrow \quad (12a)$$

$$= -\frac{B}{2m} \pm \frac{\sqrt{B^2 - 4mk}}{2m} \Rightarrow \quad (12b)$$

$$\lambda_{1,2} = -1.875 \pm j6.818. \quad (12c)$$

The roots are complex, so the system will have a damped sinusoidal step response. Let  $\sigma = -1.875$  and  $\omega = 6.818$  such that  $\lambda_{1,2} = \sigma \pm j\omega$ . The homogeneous solution is

$$v_{m_h}(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}. \quad (13)$$

In this form,  $C_1$  and  $C_2$  are complex. It is somewhat easier to deal with

$$v_{m_h}(t) = C_1 e^{\sigma t} e^{j\omega t} + C_2 e^{\sigma t} e^{-j\omega t} \quad (14a)$$

$$= e^{\sigma t} (C_1 \cos \omega t + jC_1 \sin \omega t + C_2 \cos \omega t - jC_2 \sin \omega t) \quad (14b)$$

$$= e^{\sigma t} ((C_1 + C_2) \cos \omega t + j(C_1 - C_2) \sin \omega t). \quad (14c)$$

Let  $C_3 = C_1 + C_2$  and  $C_4 = j(C_1 - C_2)$  such that

$$v_{m_h}(t) = e^{\sigma t} (C_3 \cos \omega t + C_4 \sin \omega t). \quad (15)$$

This is a decaying (because  $\sigma < 0$ ) sinusoid. A nice aspect of this new form is that  $C_3$  and  $C_4$  are real.

**9** Now, the particular solution can be found by assuming a solution of the form  $v_{m_p}(t) = K$  for  $t \geq 0$ . Substituting this into Equation 11c (with forcing  $f(t) = 1$ , we attempt to find a solution for  $K$  (that is, *determine it*):

$$(k/m)K = 1 \Rightarrow K = m/k. \quad (16)$$

Therefore,  $v_{m_p}(t) = m/k$  is a solution, and therefore the general solution is

$$v_{m_{\text{step}}}(t) = v_{m_h}(t) + v_{m_p}(t) \quad (17a)$$

$$= e^{\sigma t} (C_3 \cos \omega t + C_4 \sin \omega t) + m/k. \quad (17b)$$

This leaves the specific solution, to be found applying the initial conditions (assumed to be zero). Before we do so, however, we need the time-derivative of the  $v_{m_{\text{step}}}$ :

$$\dot{v}_{m_{\text{step}}}(t) = e^{\sigma t} ((C_3 \sigma + C_4 \omega) \cos(\omega t) + (C_4 \sigma - C_3 \omega) \sin(\omega t)). \quad (18)$$

Now, applying the initial conditions,

$$v_{m_{\text{step}}}(0) = 0 \Rightarrow \quad (19a)$$

$$C_3 = -m/k \quad (19b)$$

$$\dot{v}_{m_{\text{step}}}(0) = 0 \Rightarrow 0 = C_3 \sigma + C_4 \omega \Rightarrow \quad (19c)$$

$$C_4 = \frac{\sigma}{\omega} \cdot \frac{m}{k}. \quad (19d)$$

**10** It's good form to re-write this as a single sinusoid:

$$v_{m_{\text{step}}}(t) = v_{m_h}(t) + v_{m_p}(t) \quad (20a)$$

$$= A_1 e^{\sigma t} \cos(\omega t + \psi_1) + m/k \quad (20b)$$

where we have used [Lecture 01.2 math.trig](#) to find

$$A_1 = \sqrt{C_3^2 + C_4^2} \quad (21a)$$

$$\psi_1 = -\arctan(C_4/C_3). \quad (21b)$$

*Superposition and the derivative property*

**11** Recall that the actual forcing function is a linear combination of the input and its time-derivative. Therefore, it is expedient to re-write the time-derivative of the unit step response:

$$\dot{v}_{m_{\text{step}}}(t) = A_1 e^{\sigma t} (\sigma \cos(\omega t + \psi_1) - \omega \sin(\omega t + \psi_1)) \quad (22a)$$

$$= A_1 A_2 e^{\sigma t} \cos(\omega t + \psi_1 + \psi_2) \quad (22b)$$

where

$$A_2 = \sqrt{\sigma^2 + \omega^2} \quad (22c)$$

$$\psi_2 = -\arctan(-\omega/\sigma). \quad (22d)$$

Finally, applying superposition and the derivative rule of LTI systems,

$$v_m(t) = (B/m)\dot{v}_{m_{\text{step}}}(t) + (k/m)v_{m_{\text{step}}} \quad (23a)$$

$$= \frac{B}{m}A_1A_2e^{\sigma t} \cos(\omega t + \psi_1 + \psi_2) + \frac{k}{m}A_1e^{\sigma t} \cos(\omega t + \psi_1) + 1. \quad (23b)$$

This is the solution!

**12** It's worth plotting the response. Begin by defining the system parameters.

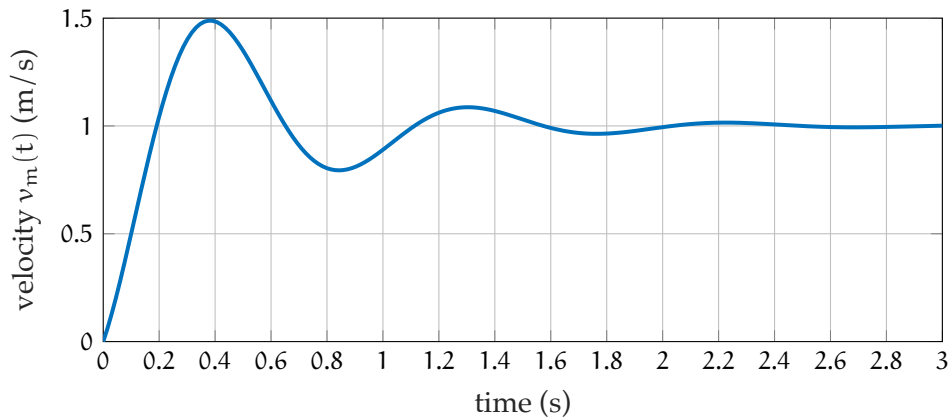
```
m = 320; % kg ... mass
k = 16000; % N/m ... spring constant
B = 1200; % N-m/s ... damping coefficient
```

Now define the secondary parameters.

```
lambda = -B/(2*m)+[-1,1]*sqrt(B^2-4*m*k)/(2*m);
sigma = real(lambda(1));
omega = imag(lambda(2));
K = m/k;
C3 = -m/k;
C4 = sigma/omega*m/k;
A1 = sqrt(C3^2+C4^2);
psi1 = -atan2(C4,C3);
A2 = sqrt(sigma^2+omega^2);
psi2 = -atan2(-omega,sigma);
```

Finally, the solution for  $v_m(t)$  can be defined as an anonymous function.

```
vm = @(t) ...
    A1*A2*B/m*exp(sigma*t).*cos(omega*t+psi1+psi2)+...
    A1*k/m*exp(sigma*t).*cos(omega*t+psi1)+...
    1;
```



**Figure vib.3:** vibration table step response  $v_m(t)$ .

Now, plot over the first few seconds. The results are shown in [Figure vib.3](#).

```
t_a = linspace(0,3,200);
h = figure;
p = plot(t_a,vm(t_a),'linewidth',1.5);
xlabel('time (s)')
ylabel('velocity $v_m(t)$ (m/s)',...
       'interpreter','latex');
grid on
hgsave(h,'figures/temp');
```

**13** Note that the steady-state output value agrees with that predicted by the equilibrium analysis, above.

## Stability

**14** We have learned what we need in order to analyze the system's stability. The roots of the characteristic equation were  $\lambda_{1,2} = -1.875 \pm j6.818$ , which clearly *all* have negative real parts, and therefore the system is *asymptotically stable*.

## 05.4 lti.ghost When gravity ghosts you

- 1 You're familiar with experience. Just when you think you're getting along so well with a "vertically" oriented translational mechanical system, gravity stops answering your texts. In this lecture, I'll try to explain this common experience: it seems that sometimes the force of gravity "matters," and other times it does not. Is gravity really Hamletic, an ambivalent vixen, or is there some way to understand this phenomenon?
- 2 Consider the following example contrived to shed some light.

### Example 05.4 lti.ghost-1

3 Often when considering a spring  $k$ , we focus on the *velocity*  $v_k$  across it, i.e. the time-derivative of the *displacement*  $x_k$ . We effectively differentiate-away the constant unstretched length  $L$  from  $x_k$ ; we can think of

$$x_k - L \quad (1)$$

as the "stretch" of the spring. In this exercise, we will attend closely to the details of this stretching.

4 Consider the mechanical harmonic oscillator shown in Fig. ghost.1. Derive a single input-output ODE for the system in terms of  $x_k$ , the total displacement across the spring. Let the constant  $\tilde{L}$  be the stretched length of the spring when the system is in static equilibrium. Solve for  $\tilde{L}$  in terms of the system parameters. Show that when we change ODE dependent variable from  $x_k$  to

$$\tilde{x}_k = x_k - \tilde{L}, \quad (2)$$

the *displacement from equilibrium*, gravity ghosts us!

5 For this example, there is a much shorter way to deriving the system ODE than our usual approach, and we will use it here: the traditional free-body diagram application of Newton's laws, shown in Fig. ghost.2. Applying

re:  
state-  
space  
model  
of a  
harmonic  
oscillator

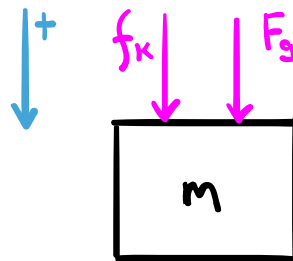
Figure ghost.1:  
a  
mechanical  
harmonic  
oscillator  
with  
spring  $k$   
unstretched  
(left)  
and  
stretched  
(right) to  
its static  
equilibrium  
length.



• Newton's second law,

where the forces are

(when  $x_k > 0$ ,  $f_k < 0$ )



**Figure ghost.2:** a free-body diagram of the mass.

Substituting in the forces,

6 Equilibrium implies  $\ddot{x}_k = 0$  and  $x_k = \tilde{L}$ , therefore

• 7 Changing variables *à la* Eq. 2 in the ODE yields



Alas, poor ghost!

8 We have seen now that the gravitational ghosting occurs when we change variables such that the displacement is relative to an *equilibrium* in the gravitational field. It is simply a change of **datum** or *reference* position of the displacement that cancels out the gravitational term—ay, there's the rub! We call this the **equilibrium requirement**.

9 For this reason, those performing such analyses, with a flourish of the hand, declare vertical displacements relative to equilibrium and poof—gravity disappears without explicit justification, for the details make cowards of us all.



**Figure ghost.3:** to ghost or not to ghost, that is the question.

10 But there are situations in which this would be a fatal error: those for which there is \_\_\_\_\_! For instance, consider if the mass in our previous example was suspended from a damper instead of a spring: in

this case, no equilibrium exists! Without going through the details or at least recalling the equilibrium requirement, it can be easy to fool oneself into wrongly dismissing gravity.

*11 Remember me,  
Ghost-would-be,  
For I am thy father's spirit,  
If gravity'd,  
With thee flee,  
Th'equilibrium requirement.*

## 05.5 lti.exe Exercises for Chapter 05

### lti

#### Exercise 05.1 oil

A certain sensor used to measure displacement over time  $t$  is tested several times with input displacement  $u_1(t)$  and a certain function  $y_1(t)$  is estimated to properly characterize the corresponding voltage output. Assuming the sensor is linear and time-invariant, what would we expect the output sensor voltage  $y_2(t)$  to be when the following input is applied?

\_\_\_\_\_/10 p.

$$u_2(t) = 3 \dot{u}_1(t) - 5 u_1(t) + \int_0^t 6 u_1(\tau) d\tau \quad (1)$$

#### Exercise 05.2 water

A system with input  $u(t)$  and output  $y(t)$  has the governing dynamical equation

\_\_\_\_\_/15 p.

$$2 \ddot{y} + 12 \dot{y} + 50 y = -10 \dot{u} + 4u. \quad (2)$$

- What is the equilibrium  $y(t)$  when  $u(t) = 6$ ?
- Demonstrate the stability, marginal stability, or instability of the system.

#### Exercise 05.3 timmychalamet

The free response of a linear system with a given set of initial conditions is  $y_{fr}$ . The forced response of the system to input  $u_1$  is  $y_{fo1}$ . The forced response of the system to input  $u_2$  is  $y_{fo2}$ . What is the (specific) response of the system to the same set of initial conditions when  $u_1(t) + u_2(t)$  is also applied? Express your answer in terms of  $y_{fr}$ ,  $y_{fo1}$ , and  $y_{fo2}$ .

\_\_\_\_\_/10 p.

\_\_\_\_\_/15 p.

**Exercise 05.4 flopugh**

Consider a linear system with state-space model matrices

$$A = \begin{bmatrix} -4 & 11 \\ 3 & -12 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix}. \quad (3)$$

For this system, respond to the following questions and imperatives.

1. What is the equilibrium state  $\bar{x}$  for input  $u(t) = 0$ ?
2. Find the corresponding input-output ODE for the system.
3. Demonstrate the asymptotic stability, marginal stability, or instability of the system from the ODE.

- 1 In this chapter, we explore the qualities of transient response—the response of the system in the interval during which initial conditions dominate.
- 2 We focus on characterizing first- and second-order linear systems; not because they're easiest (they are), but because nonlinear systems can be **linearized** about an **operating point** and because higher-order linear system responses are just *sums of first- and second-order responses*, making “everything look first- and second-order.” Well, many things, at least.
- 3 In this chapter, we primarily consider systems represented by **single-input, single-output (SISO)** ordinary differential equations (also called io ODEs)—with time  $t$ , *output*  $y$ , *input*  $u$ , **forcing function**  $f$ , constant coefficients  $a_i, b_j$ , order  $n$ , and  $m \leq n$  for  $n \in \mathbb{N}_0$ —of the form

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = f, \text{ where} \quad (1a)$$

$$f \equiv b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u. \quad (1b)$$

Note that the forcing function  $f$  is related to but distinct from the input  $u$ . This terminology proves rather important.

## 06.1 trans.char Characteristic transient responses

**1** A system's **characteristic responses** are responses to specific forcing functions—called the **singularity functions**. The reasons these are “characteristic” are:

1. the singularity functions model commonly interesting system inputs (e.g. a sudden change in the input), and so they can be said to *characterize inputs*, and
2. the ways in which the system responds to these specific functions reveal aspects of the system (e.g. how quickly it responds), so these responses can be said to *characterize systems*.

**2** Now, one may object that [Equation 1b](#) shows that a forcing function needn't look anything like an input due to its being composed of a sum of scaled copies of the input and its derivatives. Yes, but given two key properties of linear, time-invariant (LTI) systems—**superposition** and the **differentiation** property—, knowing a system's response  $y_1$  to a forcing function  $f_1$  allows us to construct its response to that input (that is,  $y_2$  for input  $u_2 = f_1$ ) as



I know.

**3** There are three singularity functions, which are now defined as piecewise functions of time  $t$ .

**4** First, the **unit impulse** or **Dirac delta** function<sup>1</sup>  $\delta$  is defined as zero everywhere except at  $t = 0$ , when it is undefined, and has unity as its integral over all time. When  $\delta$  is scaled (e.g.  $5\delta$ ), its integral scales by the same factor. This strange little beast models a sudden “spike” in the input.

<sup>1</sup>Technically,  $\delta$  is a distribution, not a function, but we use the common, confusing, comfortably couched terminology.

- 5 Second, the **unit step** function  $u_s$  is defined as zero for  $t \leq 0$  and unity for  $t > 0$ . It models a sudden change in the input. Scaling it scales the amount of change. Often, this is considered to be the gold-standard for characterizing the transient response of a system.
- 6 Third, the **unit ramp** function  $u_r$  is defined as zero for  $t \leq 0$  and  $t$  for  $t > 0$ —that is, it is linearly increasing with unity slope. It models a steadily increasing input and is probably the least useful of the singularity functions. Scaling it scales the rate of steady change.



## 06.2 trans.firsto First-order systems in transient response

1 First order systems have input-output differential equations of the form

$$\tau \frac{dy}{dt} + y = b_1 \frac{du}{dt} + b_0 u \quad (1)$$

with  $\tau \in \mathbb{R}$  called the **time constant** of the system. Systems with a single energy storage element—such as those with electrical or thermal capacitance—can be modeled as first-order.

2 The characteristic equation yields a single root  $\lambda = -1/\tau$ , so the **homogeneous solution**  $y_h$ , for constant  $\kappa \in \mathbb{R}$ , is



### Free response

3 The **free response**  $y_{fr}$  of a system is its response to initial conditions and no forcing ( $f(t) = 0$ ). This is useful for two reasons:

1. perturbations of the system from equilibrium result in free response and
2. from superposition, the free response can be added to a forced response to find the specific response:  $y(t) = y_{fr}(t) + y_{fo}(t)$ . This allows us to use tables of solutions like [Table firsto.1](#) to construct solutions for systems with nonzero initial conditions with forcing.

4 The free response is found by applying initial conditions to the homogeneous solution. With initial condition  $y(0)$ , the free response is

$$y_{fr}(t) = y(0) e^{-t/\tau}, \quad (2)$$

which begins at  $y(0)$  and decays exponentially to zero.

### Step response

5 In what follows, we develop **forced response**  $y_{fo}$  solutions, which are the *specific solution* responses of systems to given inputs and **zero initial conditions**: all initial conditions set to zero.

6 If we consider the common situation that  $b_1 = 0$  and  $u(t) = Ku_s(t)$  for some  $K \in \mathbb{R}$ , the solution to Equation 1 is

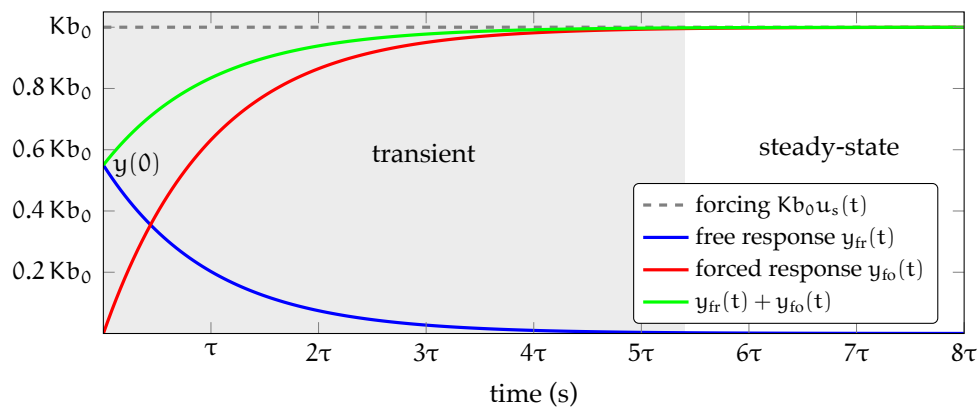


The non-steady term is simply a constant scaling of a decaying exponential.

7 A plot of the step response is shown in Figure firsto.1. As with the free response, within  $5\tau$  the transient response is less than 1% of the difference between  $y(0)$  and steady-state.

### Impulse and ramp responses

8 The response to all three singularity inputs are included in Table firsto.1. These can be combined with the free response of Equation 2 using superposition. Results could be described as **bitchin'**.



**Figure firsto.1:** free and forced responses and their sum for a first order system with input  $u(t) = Ku_s(t)$ , initial condition  $y(0)$ , and  $b_1 = 0$ .

**Table firsto.1:** first-order system characteristic and total forced responses for singularity inputs. The relevant differential equation is of the standard form  $\tau\dot{y} + y = f$ .

$u(t)$	characteristic response $f(t) = u(t)$	total forced response $y_{fo}$ for $t \geq 0$ $f(t) = b_1\dot{u} + b_0u$
$\delta(t)$	$\frac{1}{\tau}e^{-t/\tau}$	$\frac{b_1}{\tau}\delta(t) + \left(\frac{b_0}{\tau} - \frac{b_1}{\tau^2}\right)e^{-t/\tau}$
$u_s(t)$	$1 - e^{-t/\tau}$	$b_0 - \left(b_0 - \frac{b_1}{\tau}\right)e^{-t/\tau}$
$u_r(t)$	$t - \tau(1 - e^{-t/\tau})$	$b_0t + (b_1 - b_0\tau)(1 - e^{-t/\tau})$

### Example 06.2 trans.firsto-1

Consider a parallel RC-circuit with input current  $I_S(t) = 2u_s(t)$  A, initial capacitor voltage  $v_C(0) = 3$  V, resistance  $R = 1000 \Omega$ , and capacitance  $C = 1$  mF. Proceeding with the usual analysis would produce the io differential equation

$$C \frac{dv_C}{dt} + v_C/R = I_S.$$

Use Table firsto.1 to find  $v_C(t)$ .

re:  
RC-  
circuit  
response  
the  
easy  
way



## 06.3 trans.secondo Second-order systems in transient response

1 Second-order systems have input-output differential equations of the form

$$\frac{d^2y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = f(t) \quad (1)$$

where  $\omega_n$  is called the **natural frequency**,  $\zeta$  is called the (dimensionless) **damping ratio**, and  $f$  is a forcing function that depends on the input  $u$  as

$$f(t) = b_2 \frac{d^2u}{dt^2} + b_1 \frac{du}{dt} + b_0 u. \quad (2)$$

Systems with two energy storage elements—such as those with an inertial element and a spring-like element—can be modeled as second-order.

2 For distinct roots ( $\lambda_1 \neq \lambda_2$ ), the homogeneous solution is, for some real constants  $\kappa_1$  and  $\kappa_2$ ,

$$y_h(t) = \kappa_1 e^{\lambda_1 t} + \kappa_2 e^{\lambda_2 t} \quad (3)$$

where

$$\lambda_1, \lambda_2 = -\zeta\omega_n \pm \omega_n \sqrt{\zeta^2 - 1}. \quad (4)$$

### Free response

3 The free response  $y_{fr}$  is found by applying initial conditions to the homogeneous solution. With initial conditions  $y(0)$  and  $\dot{y}(0) = 0$ , the free response is

$$y_{fr}(t) = y(0) \frac{1}{\lambda_2 - \lambda_1} (\lambda_2 e^{\lambda_1 t} - \lambda_1 e^{\lambda_2 t}). \quad (5)$$

There are five possibilities for the location of the roots  $\lambda_1$  and  $\lambda_2$ , all determined by the value of  $\zeta$ .

$\zeta \in (-\infty, 0)$ : **unstable** This case is very undesirable because it means our system is unstable and, given any nonzero input or output, will **explode** to infinity.

$\zeta = 0$ : **undamped** An undamped system will oscillate forever if perturbed from zero output.

$\zeta \in (0, 1)$ : **underdamped** Roughly speaking, underdamped systems oscillate, but not forever. Let's consider the form of the solution for initial conditions and no forcing. The roots of the characteristic equation are

$$\lambda_1, \lambda_2 = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -\zeta\omega_n \pm j\omega_d \quad (6)$$

where the **damped natural frequency**  $\omega_d$  is defined as

$$\omega_d \equiv \omega_n\sqrt{1-\zeta^2}. \quad (7)$$

From Equation (5) for the free response, using Euler's formulas to write in terms of trigonometric functions, and the initial conditions  $y(0)$  and  $\dot{y}(0) = 0$ , we have

$$y_{fr}(t) = y(0) \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}} \cos(\omega_d t + \psi) \quad (8)$$

where the phase  $\psi$  is

$$\psi = -\arctan \frac{\zeta}{\sqrt{1-\zeta^2}}. \quad (9)$$

This is an oscillation that decays to the value it oscillates about,  $y(t)|_{t \rightarrow \infty} = 0$ . So any perturbation of an underdamped system will result in a decaying oscillation about equilibrium.

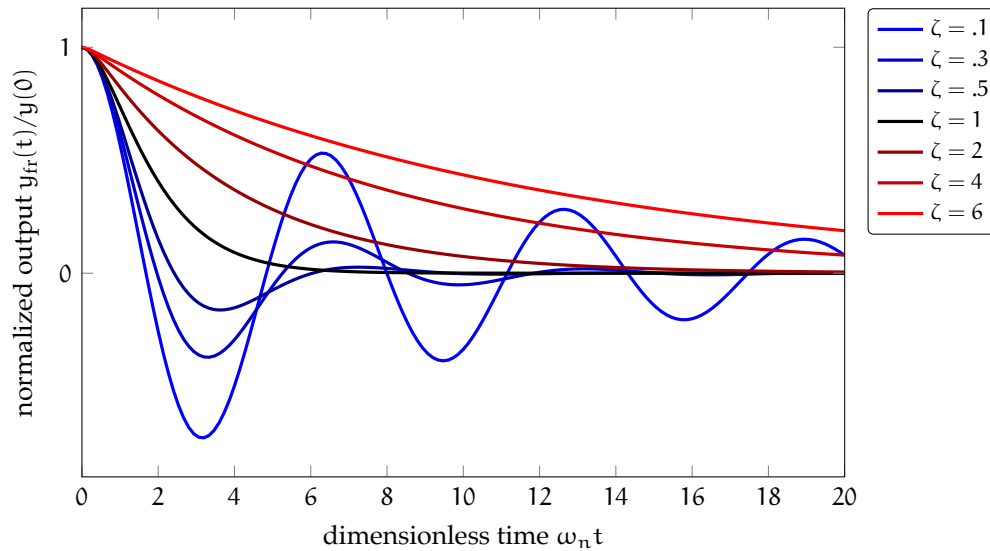
$\zeta = 1$ : **critically damped** In this case, the roots of the characteristic equation are equal:

$$\lambda_1 = \lambda_2 = -\omega_n \quad (10)$$

So we must modify Equation 3 with a factor of  $t$  for the homogeneous solution. The free response for initial conditions  $y(0)$  and  $\dot{y}(0) = 0$ , we have

$$y_{fr}(t) = y(0) (1 + \omega_n t) e^{-\omega_n t}. \quad (11)$$

This decays without oscillation, but just barely.



**Figure secondo.1:** free response  $y_{fr}(t)$  of a second-order system with initial conditions  $y(0)$  and  $\dot{y}(0) = 0$  for different values of  $\zeta$ . Underdamped, critically damped, and overdamped cases are displayed.

$\zeta \in (1, \infty)$ : **overdamped** Here the roots of the characteristic equation are distinct and real. From Equation (5) with free response to initial conditions  $y(0)$  and  $\dot{y}(0) = 0$ , we have the sum of two decaying real exponentials. This response will neither overshoot nor oscillate—like the critically damped case—but with even less gusto.

4 Figure secondo.1 displays the free response results. Note that a small damping ratio results in overshooting and oscillation about the equilibrium value. In contrast, large damping ratio results in neither overshoot nor oscillation. However, both small and large damping ratios yield responses that take longer durations to approach equilibrium than damping ratios near unity. In terms of system performances, there are tradeoffs on either side of  $\zeta = 1$ . Slightly less than one yields faster responses that overshoot a small amount. Slightly greater than one yields slower responses less prone to oscillation.

### Step response

5 Second-order systems are subjected to a variety of forcing functions  $f$ . In this lecture, we examine a common one: step forcing. In what follows, we develop **forced response**  $y_{fo}$  solutions.

6 Unit step forcing of the form  $f(t) = u_s(t)$ , where  $u_s$  is the unit step function, models abrupt changes to the input. The solution is found by applying zero initial conditions ( $y(0) = 0$  and  $\dot{y}(0) = 0$ ) to the general solution. If the roots of the characteristic equation  $\lambda_1$  and  $\lambda_2$  are distinct, the forced response is

$$y_{fo}(t) = \frac{1}{\omega_n^2} \left( 1 - \frac{1}{\lambda_2 - \lambda_1} (\lambda_2 e^{\lambda_1 t} - \lambda_1 e^{\lambda_2 t}) \right) \quad (12)$$

where

$$\lambda_1, \lambda_2 = -\zeta\omega_n \pm \omega_n \sqrt{\zeta^2 - 1}. \quad (13)$$

Once again, there are five possibilities for the location of the roots of the characteristic equation  $\lambda_1$  and  $\lambda_2$ , all determined by the value of  $\zeta$ .

However, there are three stable cases: underdamped, critically damped, and overdamped.

$\zeta \in (0, 1)$  **underdamped** In this case, the roots are distinct and complex:

$$\lambda_1, \lambda_2 = -\zeta\omega_n \pm j\omega_d. \quad (14)$$

From Equation 12, the forced step response is

$$y_{fo}(t) = \frac{1}{\omega_n^2} \left( 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1 - \zeta^2}} \cos(\omega_d t + \psi) \right) \quad (15)$$

where the phase  $\psi$  is

$$\psi = -\arctan \frac{\zeta}{\sqrt{1 - \zeta^2}}. \quad (16)$$

This response overshoots, oscillates about, and decays to  $1/\omega_n^2$ .

$\zeta = 1$  **critically damped** The roots are equal and real:

$$\lambda_1, \lambda_2 = -\omega_n \quad (17)$$



so the forced step of Equation 12 must be modified; it reduces to

$$y_{fo}(t) = \frac{1}{\omega_n^2} (1 - e^{-\omega_n t}(1 + \omega_n t)). \quad (18)$$

This response neither oscillates nor overshoots its steady-state of  $\frac{1}{\omega_n^2}$ , but just barely.

$\zeta \in (1, \infty)$  **overdamped** In this case, the roots are distinct and real, given by Equation 13. The forced step given by Equation 12 is the sum of two decaying real exponentials. These responses neither overshoot nor oscillate about their steady-state of  $1/\omega_n^2$ . With increasing  $\zeta$ , approach to steady-state slows.

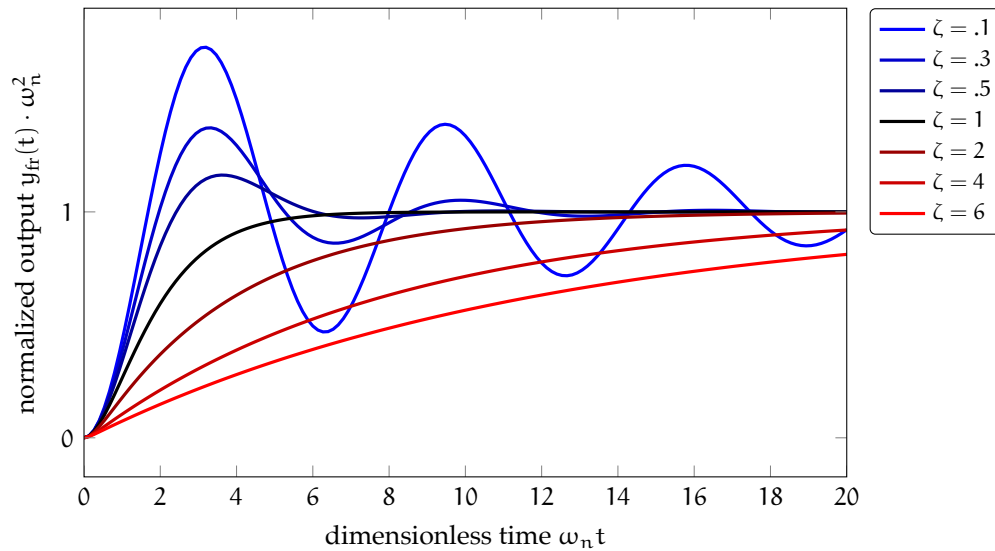
7 Figure secondo.2 displays the forced step response results. These responses are inverted versions of the free responses of Lecture 06.3 trans.secondo. Note that a small damping ratio results in overshooting and oscillation about the steady-state value. In contrast, large damping ratio results in neither overshoot nor oscillation. However, both small and large damping ratios yield responses that take longer durations to approach equilibrium than damping ratios near unity. For this reason, the damping ratio of a system should be close to  $\zeta = 1$ . There are tradeoffs on either side of one. Slightly less yields faster responses that overshoot a small amount. Slightly greater than one yields slower responses less prone to oscillation.

### Impulse and ramp responses

8 The response to all three singularity inputs are included in Table secondo.1. These can be combined with the free response of Equation 2 using superposition.

### An example with superposition

9 The results of the above are powerful not so much in themselves, but when they are wielded with the principle of superposition, as the following example shows.



**Figure secondo.2:** forced step response  $y_{fo}(t)$  of a second-order system for different values of  $\zeta$ . Underdamped, critically damped, and overdamped cases are displayed.

### Example 06.3 trans.secondo-1

In magnetic resonance force microscopy (MRFM), the primary detector is a small cantilever beam with a magnetic tip. Model the beam as a spring-mass-damper system with mass  $m = 6 \text{ pg}$ ,<sup>a</sup> spring constant  $k = 15 \text{ mN/m}$ , and damping coefficient  $B = 37.7 \cdot 10^{-15} \text{ N}\cdot\text{s/m}$ . Magnetic input forces on the beam  $u(t)$  are applied directly to the magnetic tip. That is, a Newtonian force-analysis yields the input-output ODE

$$m\ddot{y} + B\dot{y} + ky = u,$$

where  $y$  models the motion of the tip.

1. What is the natural frequency  $\omega_n$ ?
2. What is the damping ratio  $\zeta$ ?
3. Find the free response for initial conditions  $y(0) = 10 \text{ nm}$  and  $\dot{y}(0) = 0$ .
4. Find the impulse (forced) response for input  $u(t) = 3\delta(t)$ .

re:  
MRFM  
cantilever  
beam  
with  
initial  
condition  
and  
forcing

**Table second.1:** responses of system  $\frac{d^2y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = f$  to different singularity forcing. Note that  $\tau_1 = -1/\lambda_1$ ,  $\tau_2 = 1/\lambda_2$ , and  $\psi = -\arctan(\zeta/\sqrt{1-\zeta^2})$ .

damping	$f(t)$	characteristic response
$0 \leq \zeta < 1$	$\delta(t)$	$\frac{e^{-\zeta\omega_n t}}{\omega_n \sqrt{1-\zeta^2}} \sin(\omega_d t)$
	$u_s(t)$	$\frac{1}{\omega_n^2} \left( 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}} \cos(\omega_d t + \psi) \right)$
	$u_r(t)$	$\frac{1}{\omega_n^2} \left( t + \frac{e^{-\zeta\omega_n t}}{\omega_n} \left( 2\zeta \cos \omega_d t + \frac{2\zeta^2 - 1}{\sqrt{1-\zeta^2}} \sin \omega_d t \right) - \frac{2\zeta}{\omega_n} \right)$
$\zeta = 1$	$\delta(t)$	$te^{-\omega_n t}$
	$u_s(t)$	$\frac{1}{\omega_n^2} (1 - e^{-\omega_n t} - \omega_n t e^{-\omega_n t})$
	$u_r(t)$	$\frac{1}{\omega_n^2} \left( t + \frac{2}{\omega_n} e^{-\omega_n t} + t e^{-\omega_n t} - \frac{2}{\omega_n} \right)$
$\zeta > 1$	$\delta(t)$	$\frac{1}{2\omega_n \sqrt{\zeta^2 - 1}} (e^{-t/\tau_1} - e^{-t/\tau_2})$
	$u_s(t)$	$\frac{1}{\omega_n^2} \left( 1 - \frac{\omega_n}{2\sqrt{\zeta^2 - 1}} (\tau_1 e^{-t/\tau_1} - \tau_2 e^{-t/\tau_2}) \right)$
	$u_r(t)$	$\frac{1}{\omega_n^2} \left( t - \frac{2\zeta}{\omega_n} + \frac{\omega_n}{2\sqrt{\zeta^2 - 1}} (\tau_1^2 e^{-t/\tau_1} - \tau_2^2 e^{-t/\tau_2}) \right)$

5. Find the total response for the initial condition and forcing, from above, are *both* applied.

<sup>a</sup>One pg =  $10^{-12}$ g =  $10^{-15}$ kg.



## 06.4 trans.exe Exercises for Chapter 06 trans

### Exercise 06.1 Truman

Consider the i/o ODE with independent variable  $t$  and dependent variable  $y$ :

\_\_\_\_\_/25 p.

$$7\dot{y} + y = \dot{u} - 5u$$

with input

$$u(t) = u_r$$

the unit ramp function.

- What is the time constant  $\tau$ ?
- Find the *characteristic* response  $y_r$  of the system to the unit ramp input. Strongly consider using [Table firsto.1](#).
- What is the *forced* response  $y_{fo}$  to the same input?
- What is the free response of the  $y_{fr}$  to initial condition  $y(0) = 8$ ?
- What is the total response  $y_t$  when both the input  $u$  and initial condition  $y(0)$  are applied simultaneously?

### Exercise 06.2 Mogul

Consider the i/o ODE with independent variable  $t$  and dependent variable  $y$ :

$$\ddot{y} + 5\dot{y} + 25y = 2\dot{u} + 3u$$

with input

$$u(t) = u_s$$

the unit step function.

- What are the natural frequency  $\omega_n$  and damping ratio  $\zeta$ ?

- b. Find the *characteristic* response of the system to the unit step input. Stongly consider using [Table secondo.1](#).
- c. What is the *forced* response to the unit step input?

### Exercise 06.3 kibble

Consider the input-output ODE with independent variable  $t$ , dependent variable (output)  $y(t)$ , and input  $u(t)$ :

\_\_\_\_\_/ 20 p.

$$\dot{y} + 3y = 2\dot{u} + u.$$

- a. What is the time constant  $\tau$ ?
- b. Find the *characteristic* response  $y_s$  of the system to the unit step input  $u(t) = u_s(t)$ . Stongly consider using [Table firsto.1](#).
- c. What is the *forced* response  $y_{fo}$  to the input  $u(t) = 3u_s(t)$ ?
- d. What is the *free* response of the  $y_{fr}$  to initial condition  $y(0) = -4$ ?
- e. What is the total response  $y_t$  when both the input  $u$  from [Item c.](#) and initial condition  $y(0)$  are applied simultaneously?

### Exercise 06.4 biology

Consider a system with the following input-output ODE with independent variable  $t$ , dependent variable (output)  $y(t)$ , and input  $u(t)$ :

\_\_\_\_\_/ 30 p.

$$\ddot{y} + 5\dot{y} + 25y = \dot{u} + 7u$$

- a. What are the natural frequency  $\omega_n$  and damping ratio  $\zeta$ ?
- b. Find the *characteristic* response  $y_\delta$  of the system to the unit impulse forcing  $f(t) = \delta(t)$ . *Hint:* Stongly consider using [Table secondo.1](#).
- c. What is the *forced* response  $y_{fo}$  to the input  $u(t) = \delta(t)$ ?
- d. What is the *free* response  $y_{fr}$  to initial condition  $y(0) = 11$ ?
- e. What is the total response  $y_t$  when both the input  $u$  from [Item c.](#) and initial condition from [Item d.](#) are applied simultaneously?
- f. For a constant input  $u(t) = \bar{u}$ , what is the equilibrium output  $y(t) = \bar{y}$ ?
- g. Demonstrate the stability, marginal stability, or instability of the system.

## 07 ssresp

1 Recall that, for a state-space model, the state  $\mathbf{x}$ , input  $\mathbf{u}$ , and output  $\mathbf{y}$  vectors interact through two equations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (1a)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \quad (1b)$$

where  $\mathbf{f}$  and  $\mathbf{g}$  are vector-valued functions that depend on the system.

Together, they comprise what is called a **state-space model** of a system.

2 In accordance with the definition of a state-determined system, given an initial condition  $\mathbf{x}(t_0)$  and input  $\mathbf{u}$ , the state  $\mathbf{x}$  is determined for all  $t \geq t_0$ .

Determining the state response requires the solution—analytic or numerical—of the vector differential equation [Eq. 1a](#).

3 The second equation [\(1b\)](#) is *algebraic*. It expresses how the output  $\mathbf{y}$  can be constructed from the state  $\mathbf{x}$  and input  $\mathbf{u}$ . This means we must first solve the state equation [\(1a\)](#) for  $\mathbf{x}$ , then the output  $\mathbf{y}$  is given by [Eq. 1b](#).

4 Just because we know that, for a state-determined system, there *exists* a solution to [Eq. 1a](#), doesn't mean we know how to find it. In general,

$\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^n$  and  $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{R} \rightarrow \mathbb{R}^m$  can be nonlinear functions.<sup>1</sup>

We don't know how to solve most nonlinear state equations analytically. An additional complication can arise when, in addition to states and inputs, system parameters are themselves time-varying (note the explicit time  $t$  argument of  $\mathbf{f}$  and  $\mathbf{g}$ ). Fortunately, often a linear, time-invariant (LTI) model is sufficient.

<sup>1</sup>Technically, since  $\mathbf{x}$  and  $\mathbf{u}$  are themselves functions,  $\mathbf{f}$  and  $\mathbf{g}$  are *functionals*.

5 Recall that an LTI state-space model is of the form

$$\frac{dx}{dt} = Ax + Bu \quad (2a)$$

$$y = Cx + Du, \quad (2b)$$

where  $A$ ,  $B$ ,  $C$ , and  $D$  are constant matrices containing system lumped-parameters such as mass or inductance. See [Chapter 03 ss](#) for details on the derivation of such models.

6 In this chapter, we learn to solve [Eq. 2a](#) for the state response and substitute the result into [Eq. 2b](#) for the output response. First, we learn an analytic solution technique. Afterward, we learn simple software tools for numerical solution techniques.



## 07.1 `ssresp.response` Solving for the state-space response

1 In this lecture, we solve the state equation for the **state response**  $x(t)$  and substitute this into the output equation for the **output response**  $y(t)$ .

### State response

2 The state equation can be solved by a synthesis of familiar techniques, as follows. First, we rearrange:

$$\frac{dx}{dt} - Ax = Bu. \quad (1)$$

An integrating factor would be clutch, but what should it be? It looks analogous to a scalar ODE that would use the natural exponential  $\exp(-at)$  (for positive constant  $a$ ), but we have a vector ODE. We need a matrix-version of the exponential. Recall that a series definition of the scalar exponential function  $\exp : \mathbb{C} \rightarrow \mathbb{C}$  is



We define the **matrix exponential**  $\exp : \mathbb{C}^n \times \mathbb{C}^n \rightarrow \mathbb{C}^n \times \mathbb{C}^n$  (we use the same symbol) to be, for  $n \times n$  complex matrix  $Z$ ,

$$\exp Z = \sum_{k=0}^{\infty} \frac{1}{k!} Z^k. \quad (2)$$

because why not? For the hell of it, let's see if the matrix exponential

$$\exp(-At) \quad (3)$$

works as an integrating factor, if for no other reason than it was constructed to be a sort of matrix-analog of  $\exp(-at)$ , which would work for the scalar case. Premultiplying (1) on both sides:

(exercise: prove  $d \exp(-At)/dt = -\exp(-At)A$ )

Rearranging and integrating over the interval  $(0, t)$ ,

$$(\exp(0) = I)$$

This last expression can be solved for  $\mathbf{x}$ , the **state response solution**. Before we do this, however, let's define the matrix function called the **state transition matrix**  $\Phi$  to be the matrix-valued function

$$\Phi(t) = \exp(At), \quad (4)$$

Substituting  $\Phi$  and solving,

$$\mathbf{x} = \Phi(t)\mathbf{x}(0) + \Phi(t) \int_0^t \Phi(-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau \quad (5a)$$

$$= \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau. \quad (5b)$$

Note that the first term of each version of Eq. 5 is the **free response** (due to initial conditions) and the second term is the **forced response** (due to inputs).

### State transition matrix

**3** The state transition matrix  $\Phi$  introduced in Eq. 4 wound up being a key aspect of the response, which is why we call it that. We used two of its properties (in matrix exponential form) during that derivation: the

**initial-value**

$$\Phi(0) = I \quad (\text{where } I \text{ is the identity matrix})$$

and the **inverse**

$$\Phi^{-1}(t) = \Phi(-t). \quad (6)$$

**4** There is a third property that might be called the **bootstrapping property**: for time intervals  $\Delta t_i$ ,

$$\Phi(\Delta t_1 + \Delta t_2 + \dots) = \Phi(\Delta t_1)\Phi(\Delta t_2) \dots \quad (7)$$

This allows one to compute the state transition matrix<sup>2</sup> *incrementally*, from one previously computed.

**5** A final property we'll consider is the special-case of a **diagonal**  $A$  with diagonal elements  $a_{11}, a_{22}, \dots, a_{nn}$ , which yields a diagonal state transition matrix



**6** The last property turns out to be quite convenient for **deriving**  $\Phi$  for a given system, as we will see in [Lec. 07.4 ssresp.diag](#). For now, we must rely on the definition of  $\Phi$  from [Eq. 4](#) and the series definition of the matrix exponential from [Eq. 2](#). This requires us to derive the first several terms of the series solution and attempt to divine the corresponding scalar exponential series, a rather tedious task. Other than to familiarize ourselves with the definition through exercises, we prefer the derivation method of [Lec. 07.4 ssresp.diag](#).

---

<sup>2</sup>As is common, we refer to it as the “state transition matrix at a certain time,” but, technically, it’s the *image* of the state transition matrix (which is actually a matrix-valued function) at a certain time. It is good to occasionally acknowledge the violence we do to math.

### Output response

7 The output response  $\mathbf{y}(t)$  requires little additional solution: assuming we have solved for the state response  $\mathbf{x}(t)$ , the output is given in the output equation Eq. 2b. Through direct substitution, we find the **output response solution**

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \quad (8a)$$

$$= \mathbf{C}\Phi(t)\mathbf{x}(0) + \mathbf{C} \int_0^t \Phi(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t). \quad (8b)$$

## 07.2 *ssresp.eig* Linear algebraic eigenproblem

**1** The linear algebraic **eigenproblem** can be simply stated. For  $n \times n$  real matrix  $A$ ,  $n \times 1$  complex vector  $\mathbf{m}$ , and  $\lambda \in \mathbb{C}$ ,  $\mathbf{m}$  is defined as an **eigenvector** of  $A$  if and only if it is nonzero and

$$A\mathbf{m} = \lambda\mathbf{m} \quad (1)$$

for some  $\lambda$ , which is called the corresponding **eigenvalue**. That is,  $\mathbf{m}$  is an eigenvector of  $A$  if its linear transformation by  $A$  is equivalent to its scaling; i.e. an eigenvector of  $A$  is a vector of which  $A$  changes the length, but not the direction.

**2** Since a matrix can have several eigenvectors and corresponding eigenvalues, we typically index them with a subscript; e.g.  $\mathbf{m}_i$  pairs with  $\lambda_i$ .

### Solving for eigenvalues

Eq. 1 can be rearranged:

$$(\lambda I - A)\mathbf{m} = \mathbf{0}. \quad (2)$$

For a nontrivial solution for  $\mathbf{m}$ ,

$$\det(\lambda I - A) = 0, \quad (3)$$

which has as its left-hand-side a polynomial in  $\lambda$  and is called the **characteristic equation**. We define **eigenvalues** to be the roots of the characteristic equation.

#### Box 07 *ssresp.1* eigenvalues and roots of the characteristic equation

If  $A$  is taken to be the linear state-space representation  $A$ , and the state-space model is converted to an input-output differential equation, the resulting ODE's "characteristic equation" would be identical to this matrix characteristic equation. Therefore, everything we already

understand about the roots of the “characteristic equation” of an i/o ODE—especially that they govern the transient response and stability of a system—holds for a system’s  $A$ -matrix eigenvalues.

3 Here we consider only the case of  $n$  distinct eigenvalues. For eigenvalues of (algebraic) multiplicity greater than one (i.e. repeated roots), see the discussion of [Appendix 02.1 adv.eig](#).

### Solving for eigenvectors

4 Each eigenvalue  $\lambda_i$  has a corresponding eigenvector  $\mathbf{m}_i$ . Substituting each  $\lambda_i$  into [Eq. 2](#), one can solve for a corresponding eigenvector. It’s important to note that an eigenvector is unique within a scaling factor. That is, if  $\mathbf{m}_i$  is an eigenvector corresponding to  $\lambda_i$ , so is  $3\mathbf{m}_i$ .<sup>3</sup>

#### Example 07.2 [ssresp.eig-1](#)

Let

$$A = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}.$$

- Find the eigenvalues and eigenvectors of  $A$ .

**re:**  
**eigenproblem**  
**for a**  
 **$2 \times 2$**   
**matrix**

<sup>3</sup>Also of note is that  $\lambda_i$  and  $\mathbf{m}_i$  can be complex.



5 Several computational software packages can easily solve for eigenvalues and eigenvectors. See [Lec. 07.3 ssresp.eigcomp](#) for instruction for doing so in Matlab and Python.

## 07.3 `ssresp.eigcomp` Computing eigendecompositions

1 Computing eigendecompositions is rather straightforward with a numerical or symbolic computing tool such as those available in Matlab or Python. The following sections show how to use Matlab and Python to compute numerical and symbolic eigendecompositions.

### Matlab eigendecompositions

*Matlab numerical eigendecompositions*

Consider the following matrix A.

```
A = [ ...
      -3, 5, 9; ...
      0, 2, -10; ...
      5, 0, -4 ...
    ];
```

What are its eigenvalues and eigenvectors? Let's use the MATLAB function `eig`. From the documentation:

*$[V,D] = \text{EIG}(A)$  produces a diagonal matrix  $D$  of eigenvalues and a full matrix  $V$  whose columns are the corresponding eigenvectors so that  $A*V = V*D$ .*

Let's try it.

```
[Ve,De] = eig(A);
disp(vpa(Ve,3))
```

```
[ -0.769,    0.122 - 0.537i,    0.122 + 0.537i]
[  0.381,                0.767,                0.767]
[  0.514, - 0.0953 - 0.316i, - 0.0953 + 0.316i]
```

The eigenvalues are on the diagonal of `De`.



```
disp(diag(De))
```

```
-11.487 +      0i
 3.2433 +    4.122i
 3.2433 -    4.122i
```

The eigenvectors are normalized to have unit length.

```
disp(norm(Ve(:,3))) % for instance
```

```
1
```

### Matlab symbolic eigendecompositions

Sometimes symbolic parameters in a matrix require symbolic eigendecomposition. In Matlab, this requires the `symbolic` toolbox. First, declare symbolic variables.

```
syms a b c
```

Now form a symbolic matrix.

```
A = [ ...
      a,b; ...
      0,c; ...
    ]
```

```
A =
[ a, b]
[ 0, c]
```

The function `eig` is overloaded and if `A` is symbolic, the symbolic routine is called, which has a syntax similar to the numerical version above.

```
[Ve_sym,De_sym] = eig(A)
```

```
Ve_sym =  
[ 1, -b/(a - c)]  
[ 0,      1]  
De_sym =  
[ a, 0]  
[ 0, c]
```

Again, the eigenvalues are on the diagonal of the eigenvalue matrix.

```
disp(diag(De_sym))
```

```
a  
c
```

## Python eigendecompositions

*Python numerical eigendecompositions*

In Python, we first need to load the appropriate packages.

```
import numpy as np # for numerics  
from numpy import linalg as la # for eig  
from IPython.display import display, Markdown, Latex # prty  
np.set_printoptions(precision=3) # for pretty
```

Consider the same numerical  $A$  matrix from the section above. Create it as a `numpy.array` object.

```
A = np.array(  
[  
    [-3, 5, 9],  
    [0, 2, -10],  
    [5, 0, -4],  
])
```

The `numpy.linalg` module (loaded as `la`) gives us access to the `eig` function.

```
e_vals,e_vecs = la.eig(A)
print(f'e-vals: {e_vals}')
print(f'modal matrix:\n {e_vecs}')
```

```
e-vals: [-11.487+0.j      3.243+4.122j   3.243-4.122j]
modal matrix:
[[-0.769+0.j      0.122-0.537j  0.122+0.537j]
 [ 0.381+0.j      0.767+0.j    0.767-0.j   ]
 [ 0.514+0.j     -0.095-0.316j -0.095+0.316j]]
```

Note that the eigenvalues are returned as a one-dimensional array, not along the diagonal of a matrix as with Matlab.

```
print(f"the third eigenvalue is {e_vals[2]:.3e}")
```

```
the third eigenvalue is 3.243e+00-4.122e+00j
```

### *Python symbolic eigendecompositions*

We use the sympy package for symbolics.

```
import sympy as sp
```

Declare symbolic variables.

```
sp.var('a b c')
```

```
(a, b, c)
```

Define a symbolic matrix A.

```
A = sp.Matrix([
    [a,b],
    [0,c]
])
display(A)
```

$$\begin{bmatrix} a & b \\ 0 & c \end{bmatrix}$$

The `sympy.Matrix` class has methods `eigenvals` and `eigenvects`. Let's consider them in turn.

```
A.eigenvals()
```

```
{a: 1, c: 1}
```

What is returned is a dictionary with our eigenvalues as its keys and the *multiplicity* (how many) of each eigenvalue as its corresponding value. The `eigenvects` method returns even more complexly structured results.

```
A.eigenvects()
```

```
[(a, 1, [Matrix([
    [1],
    [0]])]), (c, 1, [Matrix([
    [-b/(a - c)],
    [1]])])]
```

This is a list of tuples with structure as follows.

```
(<eigenvalue>, <multiplicity>, <eigenvector>)
```

Each eigenvector is given as a list of symbolic matrices. Extracting the second eigenvector can be achieved as follows.

```
A.eigenvects()[1][2][0]
```

$$\begin{bmatrix} -\frac{b}{a-c} \\ 1 \end{bmatrix}$$

## 07.4 ssresp.diag Diagonalizing basis

1 It is useful to transform a system's state vector  $\mathbf{x}$  into a special basis that **diagonalizes**—leaves nonzero components along only the diagonal—the system's  $A$ -matrix. For systems with  $n$  distinct eigenvalues, to which we limit ourselves in this discussion,<sup>4</sup> this is always possible. In diagonalized form, it will be relatively easy to solve for the state transition matrix  $\Phi$ .

### Changing basis in the state equation

2 As with all basis transformations, the basis transformation we seek can be written

$$\mathbf{x} = P\mathbf{x}' \quad \Rightarrow \quad \mathbf{x}' = P^{-1}\mathbf{x}, \quad (1)$$

where  $P$  is the **transformation matrix**,  $\mathbf{x}$  is a representation of the state vector in the original basis, and  $\mathbf{x}'$  is a representation of the state vector in the new basis.<sup>5</sup>

3 Substituting this transformation into the standard linear state-model equations yields the model

$$\dot{\mathbf{x}}' = \underbrace{P^{-1}AP}_{A'}\mathbf{x}' + \underbrace{P^{-1}B}_{B'}\mathbf{u} \quad (2a)$$

$$\mathbf{y} = \underbrace{CP}_{C'}\mathbf{x}' + \underbrace{D}_{D'}\mathbf{u}. \quad (2b)$$

### Modal and eigenvalue matrices

4 Let a state equation have matrix  $A$  with  $n$  distinct eigenvalues ( $\lambda_i$ ) and eigenvectors ( $\mathbf{m}_i$ ). Let the **eigenvalue matrix**  $\Lambda$  be defined as

<sup>4</sup>See [Appendix 02.1 adv.eig](#) for general considerations.

<sup>5</sup>We are being a bit fast-and-loose with terminology here: a vector is an object that does not change under basis transformation, only its components and basis vectors do. However, we use the common notational and terminological abuses.

$$\Lambda = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}.$$

5 Furthermore, let the **modal matrix**  $M$  be defined as

$$M = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{m}_1 & \mathbf{m}_2 & \cdots & \mathbf{m}_n \\ | & | & & | \end{bmatrix} \quad (3)$$

### Diagonalization of the state equation

6 Let the modal matrix  $M$  be the transformation matrix for our state-model. Then<sup>6</sup>  $\mathbf{x}' = M^{-1}\mathbf{x}$ .

7 The state equation becomes

$$\dot{\mathbf{x}}' = M^{-1}AM\mathbf{x}' + M^{-1}B\mathbf{u}. \quad (4)$$

The eigenproblem implies that



That is,  $A' = \Lambda$ ! Recall that  $\Lambda$  is diagonal; therefore, we have **diagonalized** the state-space model. In full-form, the diagonalized model is

$$\dot{\mathbf{x}}' = \underbrace{\Lambda}_{A'} \mathbf{x}' + \underbrace{M^{-1}B}_{B'} \mathbf{u} \quad (5a)$$

$$\mathbf{y} = \underbrace{CM}_{C'} \mathbf{x}' + \underbrace{D}_{D'} \mathbf{u}. \quad (5b)$$

<sup>6</sup>As long as there are  $n$  distinct eigenvalues,  $M$  is invertible.

### Computing the state transition matrix

8 Recall our definition of the state transition matrix  $\Phi(t) = e^{At}$ . Directly applying this to the diagonalized system of Eq. 5,

$$\Phi'(t) = e^{At} \quad (6a)$$

$$= \begin{bmatrix} e^{\lambda_1 t} & & & 0 \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ 0 & & & e^{\lambda_n t} \end{bmatrix}. \quad (6b)$$

In this last equality, we have used the **diagonal property** of the state transition matrix, defined in [Lec. 07.1 ssresp.response](#).

9 Recall that the free response solution to the state equation is given by the initial condition and state transition matrix, so

$$\mathbf{x}'_{fr}(t) = \Phi'(t)\mathbf{x}'(0) \quad (7a)$$

$$= x'_1(0)e^{\lambda_1 t} + x'_2(0)e^{\lambda_2 t} + \dots + x'_n(0)e^{\lambda_n t} \quad (7b)$$

where the initial conditions are  $\mathbf{x}'(0) = M^{-1}\mathbf{x}(0)$ . We have completely decoupled each state's free response, one of the remarkable qualities of the diagonalized system.

10 At this point, one could simply solve the diagonalized system for  $\mathbf{x}'(t)$ , then convert the solution to the original basis with  $\mathbf{x}(t) = M\mathbf{x}'(t)$ .

11 Sometimes, we might prefer to transform the diagonalized-basis state transition matrix into the original basis. The following is a derivation of that transformation.

12 Beginning with the free response solution in the diagonalized-basis and transforming the equation into the original basis, we find an expression for the original state transition matrix, as follows.

$$\begin{aligned} \mathbf{x}'_{\text{fr}}(t) &= \Phi'(t)\mathbf{x}'(0) \Rightarrow \\ \mathbf{M}^{-1}\mathbf{x}_{\text{fr}}(t) &= \Phi'(t)\mathbf{M}^{-1}\mathbf{x}(0) \Rightarrow \\ \mathbf{x}_{\text{fr}}(t) &= \underbrace{\mathbf{M}\Phi'(t)\mathbf{M}^{-1}}_{\Phi(t)}\mathbf{x}(0). \end{aligned}$$

This last expression is just the free response solution in the original basis, so we can identify

$$\Phi(t) = \mathbf{M}\Phi'(t)\mathbf{M}^{-1}. \quad (8)$$

This is a powerful result. Eq. 8 is the preferred method of deriving the state transition matrix for a given system. The eigenvalues give  $\Phi'$  and the eigenvectors give  $\mathbf{M}$ .

#### Example 07.4 sresp.diag-1

For the state equation

$$\dot{\mathbf{x}} = \begin{bmatrix} -2 & 2 \\ 2 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \mathbf{u}$$

find the state's free response to initial condition  $\mathbf{x}(0) = \begin{bmatrix} 2 & -1 \end{bmatrix}^T$ .

re:  
state  
free  
response







## 07.5 ssresp.vibe A vibration example with two modes

1 In the following example, we explore the a mechanical vibration example, especially with regard to its modes of vibration. Both undamped and (under)damped cases are considered and we discover the effects of damping.

### Example 07.5 ssresp.vibe-1

2 Consider the system of Fig. vibe.1 in which a velocity source  $V_s$  is applied to spring  $K_1$ , which connects to mass  $m_1$ , which in turn is connected via spring  $K_2$  and damper  $B$  to mass  $m_2$  which.<sup>a</sup>

re:  
vibration  
with  
two  
modes

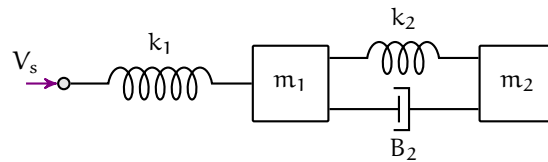


Figure vibe.1: schematic of the two-mass system.

3 The state-space model A-matrix is given as

$$A = \begin{bmatrix} -B/m_1 & -1/m_1 & B/m_1 & 0 \\ K_1 & 0 & -K_1 & 0 \\ B/m_2 & 1/m_2 & -B/m_2 & -1/m_2 \\ 0 & 0 & K_2 & 0 \end{bmatrix} \quad (1)$$

with parameters as follows:

- $m_1 = 0.1$  kg
- $m_2 = 1.1$  kg
- $K_1 = 8$  N/m

- $K_2 = 9 \text{ N/m}$

4 Two different values for  $B$  will be considered: 0 and 20 N·s/m. We will explore the modes of vibration in each case and plot a corresponding free response.

<sup>a</sup>This common situation appears in a slightly modified form in Rowell and Wormley (1997).

### Setting up the problem

We analyze the problem with Python. First, we load packages for symbolic, numeric, and graphical analysis, as follow:

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint
```

The  $A$  matrix is first defined symbolically.

```
sp.var("m_1, m_2, K_1, K_2, B", real=True)
A = sp.Matrix([
    [-B/m_1, -1/m_1, B/m_1, 0],
    [K_1, 0, -K_1, 0],
    [B/m_2, 1/m_2, -B/m_2, -1/m_2],
    [0, 0, K_2, 0]
])
```

Now define dictionaries for the parameter values.

```
p = {
    m_1: 0.1, # kg
    m_2: 1.1, # kg
    K_1: 8,   # N/m
    K_2: 9   # N/m
}
```

```
pB1 = {B: 0} # N/(rad/s), without damping
pB2 = {B: 20} # N/(rad/s), with damping
```

### Without damping

5 Without damping, we expect the system to be marginally stable and have two pairs of second-order undamped subsystems with their own unique natural frequencies. The numerical  $A$  matrix can be computed by substituting in the parameters in  $p$  and  $pB1$ , as follows:

```
A_1 = np.array(A.subs(p).subs(pB1), dtype=float)
print(A_1)
```

```
[[ 0.          -10.           0.           0.          ]
 [ 8.           0.          -8.           0.          ]
 [ 0.           0.90909091  0.          -0.90909091]
 [ 0.           0.           9.           0.          ]]
```

6 To explore the modes of vibration, we consider the eigendecomposition of  $A$ .

```
l_,M_ = np.linalg.eig(A_1)
thr = 1e-14 # threshold for calling something 0
l_.real[abs(l_.real) < thr] = 0.0 # zeroing small real parts
```

7 Let's take a closer look at the eigenvalues.

```
print(l_)
```

```
[0.+9.38179379j 0.-9.38179379j 0.+2.726993j 0.-2.726993j ]
```

8 So we have two pairs of purely imaginary eigenvalues. We would say, then, that there are two “modes of vibration,” and similarly two second-order systems comprising this fourth-order system. When we consider what the natural frequency and damping ratio is for each pair, we’re considering the natural frequencies associated with each “mode of vibration.”

9 For a second-order system (see [Lec. 06.3 trans.secondo](#)), the roots of the characteristic equation, which are equal to the eigenvalues corresponding to that second-order pair, are given in terms of natural frequency  $\omega_n$  and damping ratio  $\zeta$ :

10 So the imaginary part is nonzero only when  $\zeta \in [0, 1)$ , that is, when the system is underdamped or undamped. In this case,

(2)

11 This, taken with the fact that the eigenvalues in  $1_+$  have zero real parts, implies either  $\omega_n$  or  $\zeta$  is zero. But if  $\omega_n$  is zero, the eigenvalues would all be zero, which they are not. Therefore,  $\zeta = 0$  for both pairs of eigenvalues.

12 This leaves us with eigenvalues:

$$\pm j\omega_{n_1} \quad \text{and} \quad \pm j\omega_{n_2}. \quad (3)$$

13 So we can easily identify the natural frequencies  $\omega_{n_1}$  and  $\omega_{n_2}$  associated with each mode as follows.

```
wn_1 = np.imag(1_[0]);
wn_2 = np.imag(1_[2]);
print(f"Natural frequencies (rad/s): {wn_1} and {wn_2}")
```

```
Natural frequencies (rad/s): 9.38179378603641 and 2.726992997943728
```

*Free response*

14 Let's compute the free response to some initial conditions. The free state response is given by

15 So we can find this from the state transition matrix  $\Phi$ , which is known from Lec. 07.4 `ssresp.diag` to be \_\_\_\_\_.

16 First, we construct  $\Phi'$  symbolically.

```
sp.var("t", real=True)
L = sp.diag(*list(sp.Matrix(1_)*t)) # Eigenvalue matrix  $\Lambda$  (symbolic)
M = sp.Matrix(M_)                 # Modal matrix (symbolic)
Phi_p = sp.exp(L)
pprint(Phi_p)
```

```
Matrix([
[1.0*exp(9.38179378603641*I*t),          0,
↪ 0,                                  0],
[          0, 1.0*exp(-9.38179378603641*I*t),
↪ 0,                                  0],
[          0,          0,
↪ 1.0*exp(2.72699299794373*I*t),      0],
[          0,          0,
↪ 0, 1.0*exp(-2.72699299794373*I*t)]])
```

17 Now we can apply our transformation.

```
Phi = M*Phi_p*M.inv()
```

18 So our symbolic solution is to multiply the initial conditions by this matrix.

```
x_0 = sp.Matrix([[1], [0], [0], [0]]) # Initial condition
x = Phi*x_0                            # Free response (symbolic, messy)
```

*Plotting a free response*

19 Let's make the symbolic solution into something we can evaluate numerically and plot, a Numpy function.

```
x_fun = sp.lambdify(t,x)
```

20 Now let's set up our time array and state solution for the plot.

```
t_ = np.linspace(0,5,300)
x_ = np.squeeze(
    np.real(x_fun(t_))
)
```

**21** Plot the state responses through time. The output is shown below.

```
fig, ax = plt.subplots()
ax.plot(t_, x_.T)
ax.set_xlabel('time (s)')
ax.set_ylabel('state free response')
ax.legend(['$x_1$', '$x_2$', '$x_3$', '$x_4$'])
```

```
<matplotlib.legend.Legend at 0x127e64e30>
```



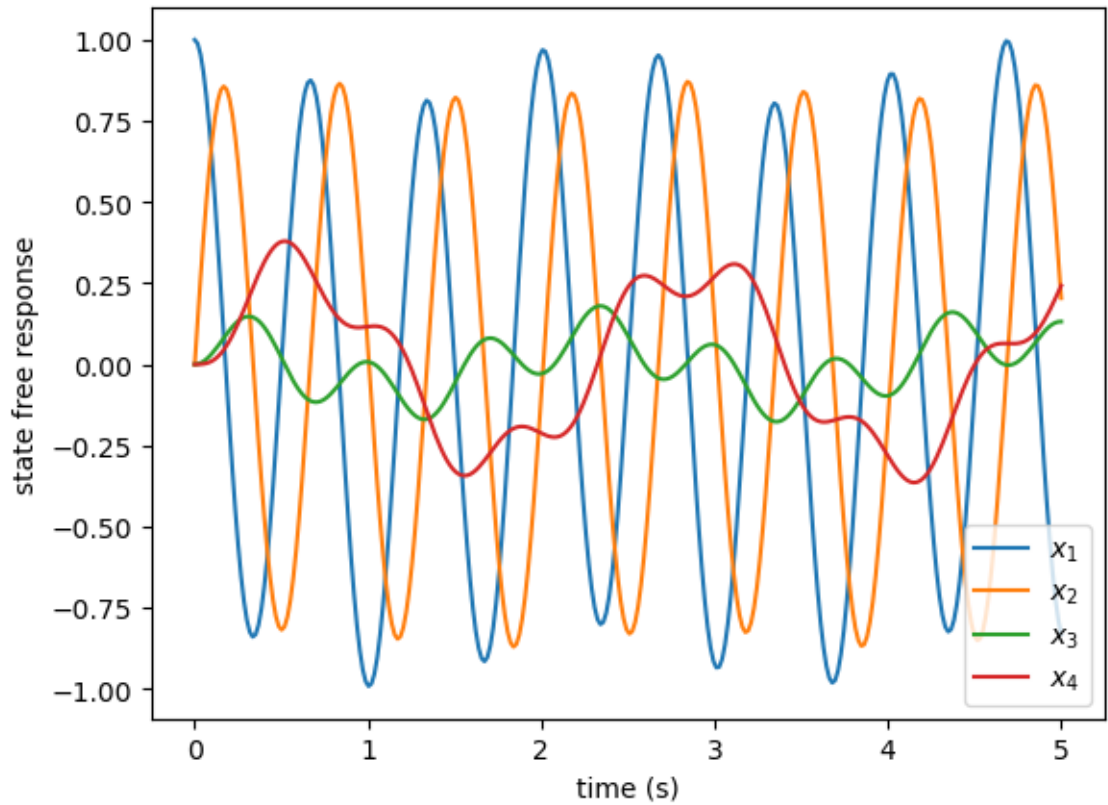


Figure vibe.2: png

### With a little damping

**22** Now consider the case when the damping coefficient  $B$  is nonzero. Let's recompute  $A$  and the eigendecomposition.

```
A_2 = np.array(A.subs(p).subs(pB2), dtype=float)
print(A_2)
```

```
[[-200.      -10.      200.         0.        ]
 [  8.         0.        -8.         0.        ]
 [ 18.18181818  0.90909091 -18.18181818 -0.90909091]
 [  0.         0.         9.         0.        ]]
```

23 To explore the modes of vibration, we consider the eigendecomposition of  $A$ .

```
l_,M_ = np.linalg.eig(A_2)
```

24 Let's take a closer look at the eigenvalues.

```
print(l_)
```

```
[-2.17777946e+02+0.j          -1.53514941e-03+2.73840736j
 -1.53514941e-03-2.73840736j  -4.00801807e-01+0.j          ]
```

25 We can see that one of the second-order systems is now “overdamped” or, equivalently, has split into two first-order systems. The other is now underdamped (but barely damped). Let's compute the natural frequency of the remaining vibratory mode.

```
wn_1 = np.imag(l_[1]);
print(f"Natural frequency (rad/s): {wn_1}")
```

```
Natural frequency (rad/s): 2.7384073593287575
```

26 So the effect of damping was to eliminate the  $\approx 10$  rad/s mode and leave us with a slightly modified version of the  $\approx 2.7$  rad/s mode.

*Free response*

27 Let's compute the free response to some initial conditions. The free state response is given by

28 So we can find this from the state transition matrix  $\Phi$ , which is known from Lec. 07.4 `ssresp.diag` to be \_\_\_\_\_.

29 First, we construct  $\Phi'$  symbolically.

```

L = sp.diag(*list(sp.Matrix(1_)*t)) # Eigenvalue matrix  $\Lambda$  (symbolic)
M = sp.Matrix(M_)                  # Modal matrix (symbolic)
Phi_p = sp.exp(L)
pprint(Phi_p)

```

```

Matrix([
[1.0*exp(-217.777946076145*t),
↪ 0,                                0,
↪ 0],
[
↪ 0, 1.0*exp(t*(-0.00153514941381959 +
↪ 2.73840735932876*I)),
↪ 0,                                0],
[
↪ 0, 1.0*exp(t*(-0.00153514941381959 - 2.73840735932876*I)),
↪ 0],
[
↪ 0,                                0,
↪ 0,                                0,
↪ 1.0*exp(-0.400801806845378*t)]]

```

**30** Now we can apply our transformation.

```
Phi = M*Phi_p*M.inv()
```

**31** So our symbolic solution is to multiply the initial conditions by this matrix.

```

x_0 = sp.Matrix([[1], [0], [0], [0]]) # Initial condition
x = Phi*x_0                            # Free response (symbolic, messy)

```

*Plotting a free response*

**32** Let's make the symbolic solution into something we can evaluate numerically and plot, a Numpy function.

```
x_fun = sp.lambdify(t,x)
```

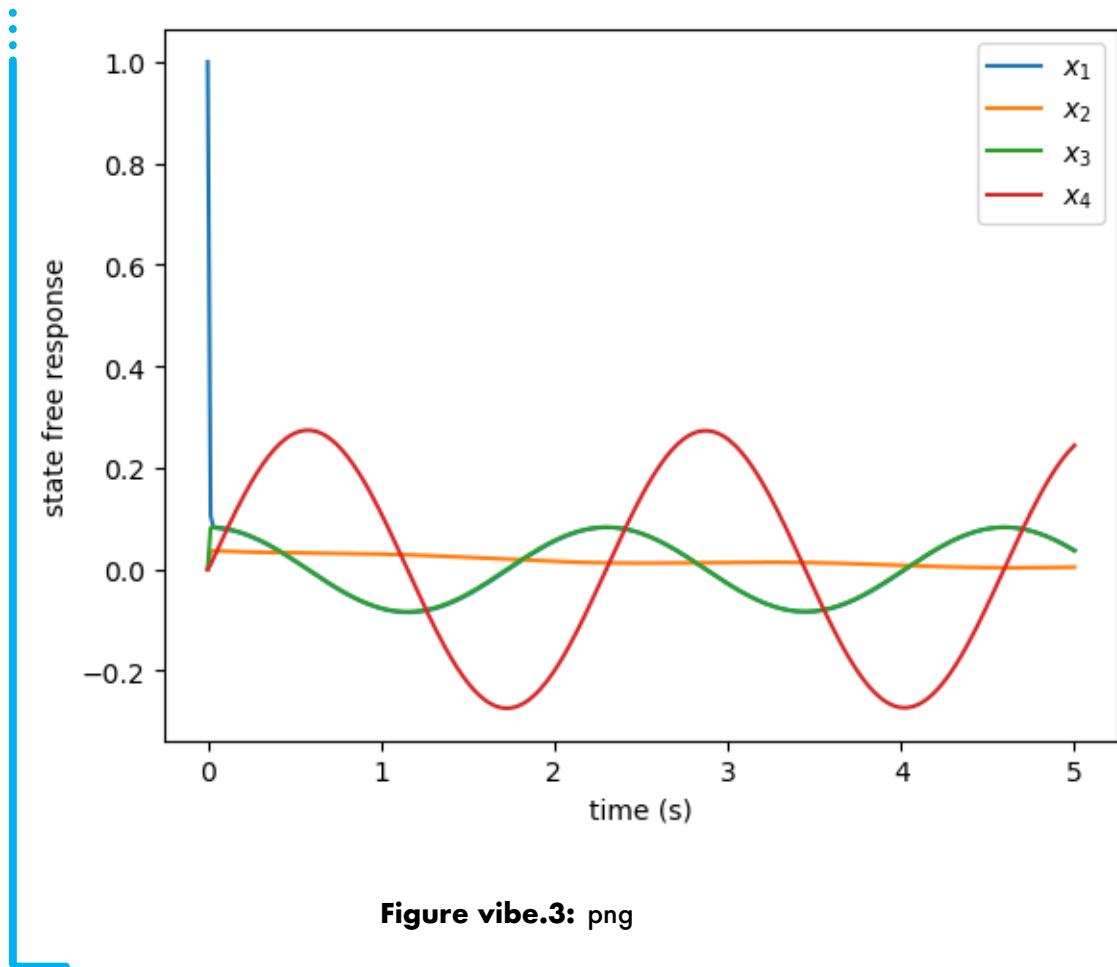
**33** Now let's set up our time array and state solution for the plot.

```
t_ = np.linspace(0,5,300)
x_ = np.squeeze(
    np.real(x_fun(t_))
)
```

**34** Plot the state responses through time. The output is shown below.

```
fig, ax = plt.subplots()
ax.plot(t_, x_.T)
ax.set_xlabel('time (s)')
ax.set_ylabel('state free response')
ax.legend(['$x_1$', '$x_2$', '$x_3$', '$x_4$'])
```

```
<matplotlib.legend.Legend at 0x137a6acf0>
```



## 07.6 `ssresp.mixed` Analytic and numerical output response example in Matlab

1 In the following example, we explore the output response derived both analytically and numerically in Matlab.

### Example 07.6 `ssresp.mixed-1`

Consider a state-space model with the following standard matrices.

```
A = [...  
  -1, 3, 5, 7;...  
   0, -2, 0, 6;...  
  -2, 1, -3, 0;...  
   0, 1, 3, -4;...  
];  
n = length(A); % order  
B = [...  
  0; 1; 0; 2; ...  
];  
C = eye(n);  
D = zeros([n,1]);
```

Solve for the unit step response output  $y$  given the following initial condition.

```
x0 = [2;0;2;0];
```

re:  
analytic  
and  
numerical  
output  
response  
solution  
in  
Matlab

### Analytic solution

We use the solution of Eq. 8:

$$\mathbf{y}(t) = \mathbf{C}\Phi(t)\mathbf{x}(0) + \mathbf{C} \int_0^t \Phi(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t). \quad (1)$$

First we need  $\Phi(t)$ . The “primed” basis requires the eigendecomposition.

```
[M,L] = eig(A);
```

We can find  $\Phi$  from the primed-basis version  $\Phi'$ , which is easy to compute.

```
Phi_p = @(t) diag(diag(exp(L*t)));
```

Now the basis transformation.

```
M_inv = M^-1; % compute just once, not on every call  
Phi = @(t) M*Phi_p(t)*M_inv;
```

Declare symbolic variables.

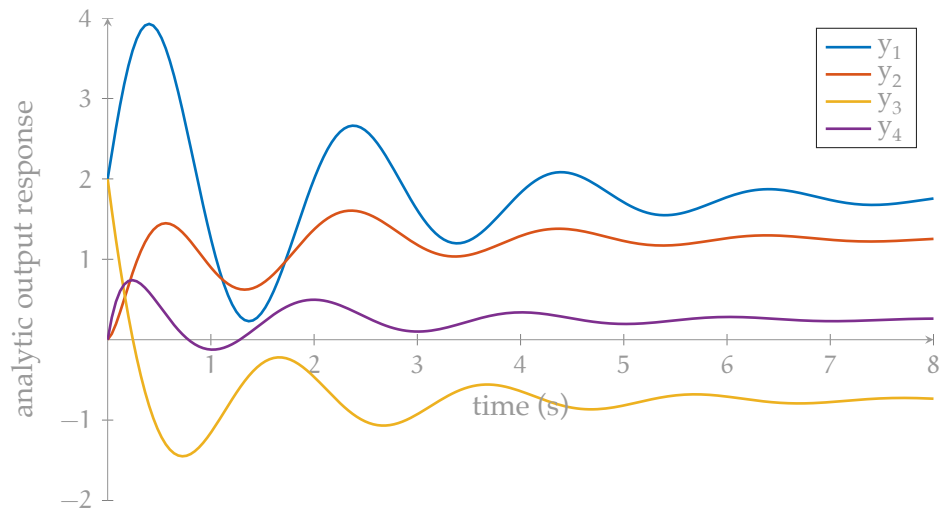
```
syms T tt
```

Apply Eq. 8.

```
y_sym = C*Phi(tt)*x0 + C*int(Phi(tt-T)*B*1,T,0,tt) + D*1;
```

Convert this to a numerically evaluable function.

```
y_num = matlabFunction(y_sym);
```



**Figure mixed.1:** the analytic output response.

Plot it; the result is shown in Fig. mixed.1.

```
figure
t_num = linspace(0,8,200);
plot(t_num,y_num(t_num),'linewidth',1)
xlabel('time (s)')
ylabel('analytic output response')
legend('y_1','y_2','y_3','y_4')
```

### Numerical solution

```
sys = ss(A,B,C,D);
```

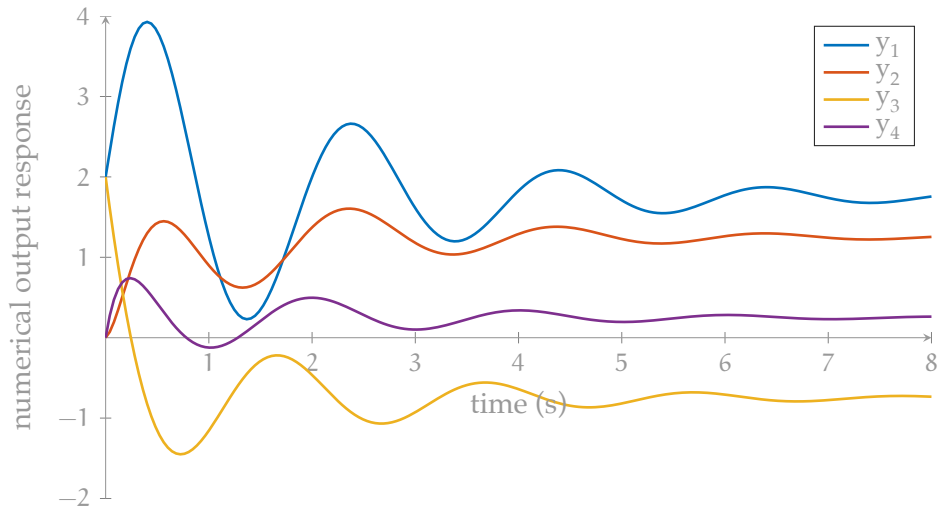
#### Using *lsim*

First, use *lsim* to compute the response numerically.

```
u_s = ones(size(t_num)); % a one for every time
y_lsim = lsim(sys,u_s,t_num,x0); % simulate
```

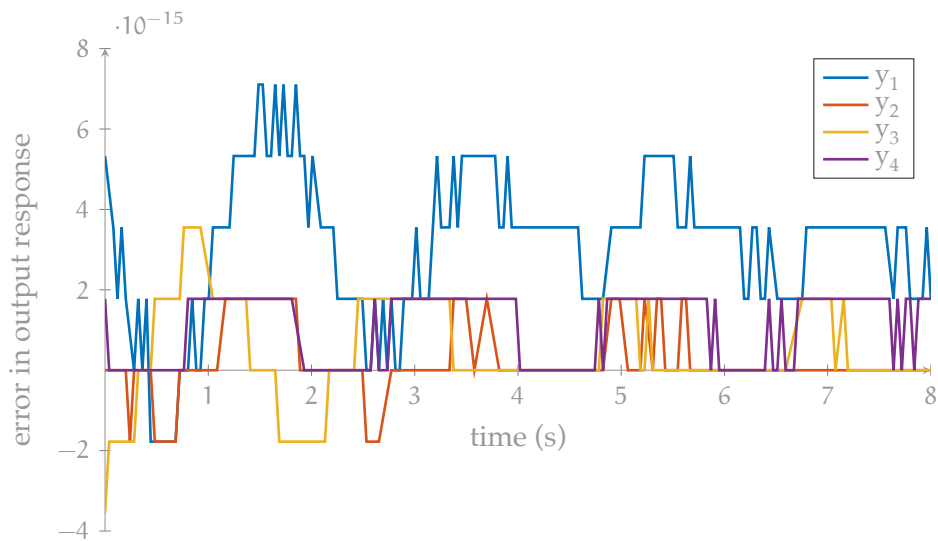
• Now plot it; the result is shown in Fig. mixed.2.





**Figure mixed.2:** numerical (using `lsim`) output response.

```
figure
plot(t_num,y_lsim,'linewidth',1)
xlabel('time (s)')
ylabel('numerical output response')
legend('y_1','y_2','y_3','y_4')
hgsave(h,'figures/temp');
```



**Figure mixed.3:** comparison of analytic and numerical output responses.

- Now take the difference between the two solutions and plot the error. As Fig. mixed.3 shows, the differences are minimal.

```
figure
plot(t_num,y_lsim-y_num(t_num) .', 'linewidth',1)
xlabel('time (s)')
ylabel('error in output response')
legend('y_1','y_2','y_3','y_4')
```

*Using the step and initial commands with superposition*

Just for fun, here's how we could use `step` and `initial` (instead of `lsim`) with superposition to numerically solve.

```
y_step = step(sys,t_num); % forced response
y_initial = initial(sys,x0,t_num); % free response
y_total = y_initial + y_step; % (superposition)
```

## 07.7 `ssresp.sim` Simulating state-space response

### 1 Ahem.<sup>7</sup>

For many nonlinear models, numerical solution of the state equation is required. For linear models, we can always solve them analytically using the methods of this chapter. However, due to its convenience, we will often want to use numerical techniques even when analytic ones are available. Matlab has several built-in and *Control Systems Toolbox* functions for analyzing state-space system models, especially *linear* models. We'll explore a few, here.

Consider, for instance, a linear state model with the following A, B, C, and D matrices:

$$A = \begin{bmatrix} -3 & 4 & 5 \\ 0 & -2 & 3 \\ 0 & -6 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (1a)$$

```
A = [-3,4,5;0,-2,3;0,-6,1];
B = [1;0;1];
C = [1,0,0;0,-1,0];
D = [0;0];
```

For a step input  $u(t) = 3u_s(t)$  and initial state  $x(0) = [1 \ 2 \ 3]^T$ , let's compare analytic and numerical solutions for the output response  $y(t)$ .

```
u = @(t) 3*ones(size(t)); % for t>=0
x_0 = [1; 2; 3];
```

<sup>7</sup>The source of this lecture can be downloaded as a Matlab m-file at [http://ricopic.one/dynamic\\_systems/source/simulating\\_state\\_space\\_response.m](http://ricopic.one/dynamic_systems/source/simulating_state_space_response.m).

### Analytic solution

For an analytic solution, we'll use a rearranged version of ??.<sup>8</sup>

$$\mathbf{y}(t) = \mathbf{C}\Phi(t)\mathbf{x}(0) + \mathbf{C}\Phi(t) \int_0^t \Phi(-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t). \quad (2a)$$

First, we need the state transition matrix  $\Phi(t)$ , so we consider the eigenproblem.

$$[\mathbf{M}, \mathbf{L}] = \text{eig}(\mathbf{A})$$

$\mathbf{M} =$

$$\begin{bmatrix} 1.0000 + 0.0000i & 0.7522 + 0.0000i & 0.7522 + 0.0000i \\ 0.0000 + 0.0000i & 0.3717 + 0.0810i & 0.3717 - 0.0810i \\ 0.0000 + 0.0000i & 0.0787 + 0.5322i & 0.0787 - 0.5322i \end{bmatrix}$$

$\mathbf{L} =$

$$\begin{bmatrix} -3.0000 + 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i \\ 0.0000 + 0.0000i & -0.5000 + 3.9686i & 0.0000 + 0.0000i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & -0.5000 - 3.9686i \end{bmatrix}$$

Note that, when assigning its output to two variables  $\mathbf{M}$  and  $\mathbf{L}$ , the `eig` function returns the modal matrix to  $\mathbf{M}$  and the eigenvalue matrix to  $\mathbf{L}$ . The modal matrix of eigenvectors  $\mathbf{M}$  has each column (eigenvector) normalized to unity. Also notice that  $\mathbf{M}$  and  $\mathbf{L}$  are *complex*. The imaginary parts of two eigenvalues and their corresponding eigenvectors are significant. Finally, since the *real* parts of the all eigenvalues are *negative*, the system is stable. The “diagonal”-basis state transition matrix  $\Phi'(t)$  is simply

$$\Phi'(t) = e^{\mathbf{L}t}. \quad (3)$$

Let's define this as an “anonymous” function.

<sup>8</sup>Although we call this the “analytic” solution, we are not solving for a detailed symbolic expression, although we *could*. In fact, Eq. 2 *is* the analytic solution and what follows is an attempt to represent it graphically.

```
Phi_p = @(t) diag(diag(exp(L*t))); % diags to get diagonal mat
```

The original-basis state transition matrix  $\Phi(t)$  is, from ??,

$$\Phi(t) = M\Phi'(t)M^{-1}. \quad (4)$$

```
M_inv = M^-1; % compute just once, not on every call
Phi = @(t) M*Phi_p(t)*M_inv;
```

*Free response*

The free response is relatively straightforward to compute.

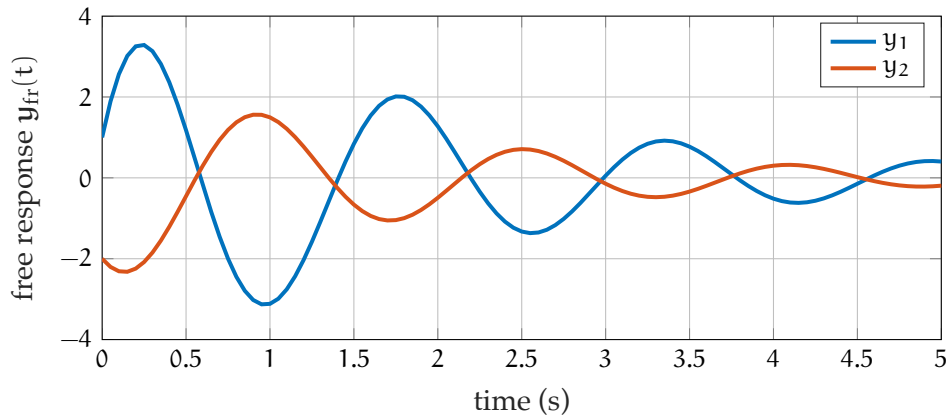
```
t_a = 0:.05:5; % simulation time
y_fr = NaN*ones(size(C,1),length(t_a)); % initialize
for i = 1:length(t_a)
    y_fr(:,i) = C*Phi(t_a(i))*x_0;
end
y_fr(:,1:3) % first three columns
```

```
ans =
    1.0000 - 0.0000i    1.8922 - 0.0000i    2.5646 - 0.0000i
   -2.0000 + 0.0000i   -2.2030 + 0.0000i   -2.3105 + 0.0000i
```

A time array  $t_a$  was defined such that  $\Phi$  could be evaluated. The first three columns of  $y_{fr}$  are printed for the first three moments in time. Note how there's a "hanging chad" of imaginary components. Before we realize them, let's make sure they're negligibly tiny.

```
max(max(abs(imag(y_fr))))
y_fr = real(y_fr);
```

```
ans =
    5.2907e-16
```



**Figure sim.1:** free response  $y_{fr}$ .

The results are plotted in Fig. sim.1. As we might expect from the eigenvalues, the free responses of both outputs oscillate and decay.

#### *Forced response*

Now, there is the matter of integration in Eq. 2. Since Matlab does not excel in symbolic manipulation, we have chosen to avoid attempting to write the solution, symbolically.<sup>9</sup> For this reason, we choose a simple numerical (trapezoidal) approximation of the integral using the trapz function. First, the integrand can be evaluated over the simulation interval.

```

integrand_a = NaN*ones(size(C,2),length(t_a)); % initialize
for i = 1:length(t_a)
    tau = t_a(i);
    integrand_a(:,i) = Phi(-tau)*B*u(tau);
end

```

Now, numerically integrate.

```

integral_a = zeros(size(integrand_a));
for i = 2:length(t_a)
    i_up = i; % upper limit of integration

```

<sup>9</sup>Mathematica or SageMath would be preferable for this.

```

integral_a(:,i) = ... % transposes for trapz
    trapz(t_a(1:i_up)',integrand_a(:,1:i_up)')';
end

```

Now, evaluate the forced response at each time.

```

y_fo = NaN*ones(size(C,1),length(t_a)); % initialize
for i = 1:length(t_a)
    y_fo(:,i) = C*Phi(t_a(i))*integral_a(:,i);
end
y_fo(:,1:3) % first three columns

```

```

ans =

    0.0000 + 0.0000i    0.1583 - 0.0000i    0.3342 - 0.0000i
    0.0000 + 0.0000i   -0.0109 + 0.0000i   -0.0426 + 0.0000i

```

```

max(max(abs(imag(y_fo))))
y_fo = real(y_fo);

```

```

ans =

    2.1409e-16

```

The forced response is shown in Fig. sim.2, which shows damped oscillations.

#### Total response

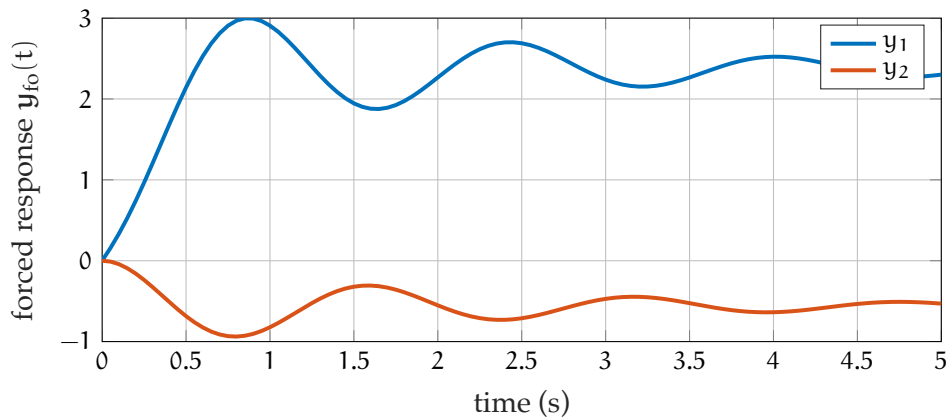
The total response is found from the sum of the free and forced responses:  $\mathbf{y}(t) = \mathbf{y}_{fr} + \mathbf{y}_{fo}$ . We can simply sum the arrays.

```

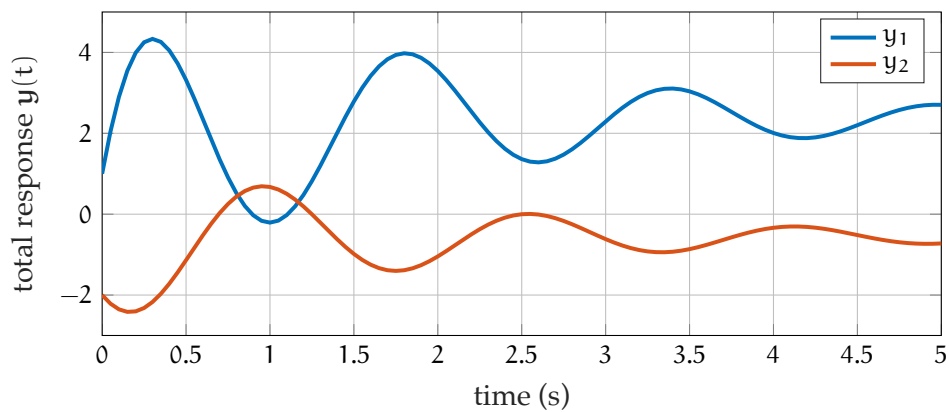
y_t = y_fr + y_fo;

```

The result is plotted in Fig. sim.3.



**Figure sim.2:** forced response  $y_{fo}$ .



**Figure sim.3:** total response  $y$ .

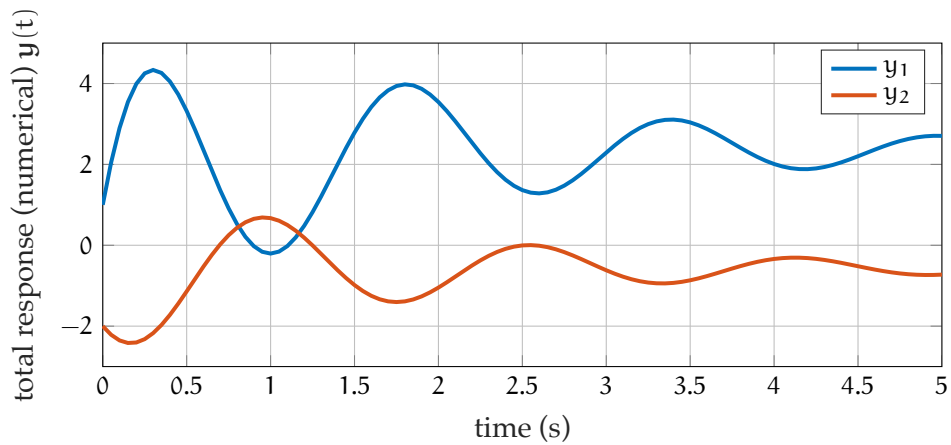
### Numerical solution

The numerical solution of the state equations is rather simple using Matlab's `ss` and `step` or `lsim` commands, as we show, here. First, we define an `ss` model object—a special kind of object that encodes a state-space model.

```
sys = ss(A,B,C,D);
```

At this point, using the step function would be the easiest way to solve for the step response. However, we choose the more-general `lsim` for





**Figure sim.4:** total response  $y$  from `lsim`.

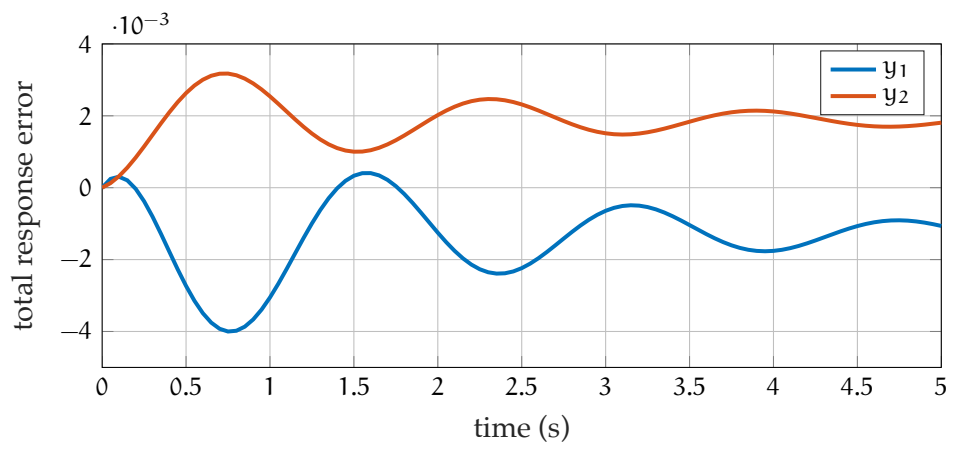
demonstration purposes.

```
y_t_num = lsim(sys,u(t_a),t_a,x_0);
```

This total solution is shown in [Fig. sim.4](#).

```
d_y = y_t-y_t_num';
```

[Fig. sim.5](#) shows a plot of the differences between the analytic total solution  $y_t$  and the numerical  $y_t_{num}$  for each output. Note that calling this “error” is a bit presumptuous, given that we used numerical integration in the analytic solution. If a more accurate method is desired, working out the solution, symbolically, is the best.



**Figure sim.5:** total response error  $y_t - y_{t\_num}$ .

## 07.8 ssresp.exe Exercises for Chapter 07 ssresp

### Exercise 07.1 larry

Let a system have the following state and output equation matrices:

$$A = \begin{bmatrix} -3 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = [0 \quad 1] \quad D = [0].$$

For this system, answer the following imperatives.

- Find the *eigenvalue matrix*  $\Lambda$  and comment on the stability of the system (justify your comment). Use the convention that  $\lambda_1 \geq \lambda_2$  and order  $\Lambda$  accordingly.
- Find the eigenvectors and the *modal matrix*  $M$ .
- Find the *state transition matrix*  $\Phi(t)$ . Hint: first find the “diagonalized” state transition matrix  $\Phi'(t)$ .
- Using the state transition matrix, find the *output free response* for initial condition

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

### Exercise 07.2 mo

Let a system have the following state and output equation matrices:

$$A = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad C = [1 \quad 0] \quad D = [0].$$

For this system, answer the following imperatives.

- Find the *eigenvalue matrix*  $\Lambda$  and comment on the stability of the system (justify your comment). Use the convention that  $\lambda_1 \leq \lambda_2$  and order  $\Lambda$  accordingly.
- Find the eigenvectors and the *modal matrix*  $M$ .

- c. Find the *state transition matrix*  $\Phi(t)$ . Hint: first find the “diagonalized” state transition matrix  $\Phi'(t)$ .
- d. Using the state transition matrix, find the *output homogeneous solution* for initial condition

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

### Exercise 07.3 curly

Use a computer for this exercise. Let a system have the following state A-matrix:

$$A = \begin{bmatrix} -2 & 2 & 0 \\ -1 & -2 & 2 \\ 0 & -1 & -2 \end{bmatrix}.$$

For this system, answer the following imperatives.

- a. Find the *eigenvalue matrix*  $\Lambda$  and *modal matrix*  $M$ .
- b. Comment on the stability of the system (justify your comment).
- c. Find the *diagonalized state transition matrix*  $\Phi'(t)$ . Be sure to print the expression. Furthermore, find the *state transition matrix*  $\Phi(t)$ .
- d. Using the state transition matrix, find the *state free response* for initial condition

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Do *not* print this expression.

- e. Plot the free response found above for  $t \in [0, 4]$  seconds.

**Exercise 07.4 lonely**

Use a computer for this exercise. Let a system have the following state and output equation matrices:

$$A = \begin{bmatrix} -1 & 0 & 8 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 2 \\ 3 & 0 \\ 0 & 0 \end{bmatrix} \quad C = [1 \ 0 \ -1] \quad D = [0 \ 0].$$

For this system, answer the following imperatives.

- Find the *eigenvalue matrix*  $\Lambda$  and comment on the stability of the system (justify your comment). Use the convention that  $\lambda_1 \geq \lambda_2 \geq \lambda_3$  and order  $\Lambda$  accordingly.
- Find the eigenvectors and the *modal matrix*  $M$ .
- Find the *state transition matrix*  $\Phi(t)$ . Hint: first find the “diagonalized” state transition matrix  $\Phi'(t)$ .
- Let the input be

$$\mathbf{u}(t) = \begin{bmatrix} 4 \\ \sin(2\pi t) \end{bmatrix}.$$

Solve for the forced *state* response  $\mathbf{x}_{fo}(t)$ . Express it simply—it’s not that bad.

- Solve for the forced *output* response  $\mathbf{y}_{fo}(t)$ . Express it simply—it’s not that bad.
- Plot  $\mathbf{y}_{fo}(t)$  for  $t \in [0, 7]$  sec.

**Exercise 07.5 artemis**

Use a computer for this exercise. Let a system have the following state and output equation matrices:

$$A = \begin{bmatrix} -5 & 6 \\ 1 & -10 \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad C = [0 \ 1], \quad D = [0].$$

For this system, answer the following imperatives.

- Find the *eigenvalue matrix*  $\Lambda$  and comment on the stability of the system (justify your comment). Use the convention that  $\lambda_1 \geq \lambda_2$  and order  $\Lambda$  accordingly.
- Find the eigenvectors and the *modal matrix*  $M$ .
- Find the *state transition matrix*  $\Phi(t)$ . Hint: first find the “diagonalized” state transition matrix  $\Phi'(t)$ .
- Let the input be

$$\mathbf{u}(t) = [\delta(t)],$$

where  $\delta$  is the Dirac delta impulse function.<sup>10</sup> Solve for the forced *state* response  $\mathbf{x}_{fo}(t)$ . Express it simply.<sup>11</sup>

- Solve for the forced *output* response  $\mathbf{y}_{fo}(t)$ . Express it simply.
- Plot  $\mathbf{y}_{fo}(t)$  for  $t \in [0, 3]$  sec.

### Exercise 07.6 level

Use a computer for this exercise. Let a system have the following state and output equation matrices:

\_\_\_\_\_/  
35 p.

$$A = \begin{bmatrix} -4 & -3 & 0 \\ 0 & -8 & 4 \\ 0 & 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad C = [0 \quad 1 \quad 0], \quad D = [0].$$

For this system, answer the following imperatives.

- Find the *eigenvalue matrix*  $\Lambda$  and comment on the stability of the system (justify your comment).
- Find the eigenvectors and the *modal matrix*  $M$ .
- Find the *state transition matrix*  $\Phi(t)$ . Hint: first find the “diagonalized” state transition matrix  $\Phi'(t)$ .

<sup>10</sup>Matlab's `dirac` can be used, symbolically. Symbolic integration yield a result with the `heaviside` function. With the setting `sympref('HeavisideAtOrigin', 0)` this is equivalent with our definition of the unit step function  $u_s$ .

<sup>11</sup>Matlab's `simplify` function may need some help. Use the `'assume'` function.

d. Let the input be

$$\mathbf{u}(t) = \begin{bmatrix} u_s(t) \end{bmatrix},$$

where  $u_s$  is the unit step function.<sup>12</sup> Solve for the forced *state* response  $\mathbf{x}_{fo}(t)$ . Express it simply.

e. Solve for the forced *output* response  $\mathbf{y}_{fo}(t)$ . Express it simply.

f. Plot  $\mathbf{y}_{fo}(t)$  for  $t \in [0, 5]$  sec.

---

<sup>12</sup>In Python, we can define a symbolic unit step function using the `sympy.Heaviside()` function. Alternatively, we can set  $u_s(t)$  equal to 1 for integration over the interval  $[0, t]$ .

## **Part III**

# **Modeling other systems**



## 08 thermoflu

### Lumped-par

- 1** We now consider the **lumped-parameter modeling** of **fluid systems** and **thermal systems**. The linear graph-based, state-space modeling techniques of [Chapters 02 graphs](#) to [04 emech](#) are called back up to service for this purpose. Recall that this method defines several types of discrete elements in an energy domain—in [Chapters 02 graphs](#) and [03 ss](#), the electrical and mechanical energy domains. Also recall from [Chapter 04 emech](#) that energy transducing elements allow energy to flow among domains. In this chapter, we introduce fluid and thermal energy domains and discrete and transducing elements associated therewith.
- 2** The analogs between the mechanical and electrical systems from [Chapter 02 graphs](#) are expanded to include fluid and thermal systems. This generalization allows us to include, in addition to electromechanical systems, inter-domain systems including electrical, mechanical, fluid, and thermal systems.
- 3** This chapter begins by defining discrete lumped-parameter elements for fluid and thermal systems. We then categorize these into energy source, energy storage (A-type and T-type), and energy dissipative (D-type) elements, allowing us to immediately construct linear graphs and normal trees in the manner of [Chapter 02 graphs](#). Then we can directly apply the methods of [Chapter 03 ss](#) to construct state-space models of systems that include fluid and thermal elements.

## 08.1 thermoflu.flu Fluid system elements

- 1 Detailed *distributed* models of fluids, such as the Navier-Stokes equations, are necessary for understanding many aspects of fluid systems and for guiding their design (e.g. a pump or an underwater vehicle). However, a great many fluid systems are networks of pipes, tanks, pumps, valves, orifices, and elevation changes—and at this *system-level*, a different approach is required.
- 2 As with electrical and mechanical systems, we can describe fluid systems as consisting of discrete lumped-parameter elements. The dynamic models that can be developed from considering these elements are often precisely the right granularity for system-level design.
- 3 We now introduce a few lumped-parameter elements for modeling fluid systems. Let a **volumetric flowrate**  $Q$  and **pressure drop**  $P$  be input to a port in a fluid element. Since, for fluid systems, the power into the element is

$$\mathcal{P}(t) = Q(t)P(t) \quad (1)$$

we call  $Q$  and  $P$  the **power-flow variables** for fluid systems. A fluid element has two distinct locations between which its pressure drop is defined. We call a reference pressure **ground**.

- 4 **Work** done on the system over the time interval  $[0, T]$  is defined as

$$W \equiv \int_0^T \mathcal{P}(\tau) d\tau. \quad (2)$$

Therefore, the work done on a fluid system is

$$W = \int_0^T Q(\tau)P(\tau) d\tau. \quad (3)$$

- 5 The **pressure momentum**  $\Gamma$  is

$$\Gamma(t) = \int_0^t P(\tau) d\tau + \Gamma(0). \quad (4)$$

Similarly, the **volume** is

$$V(t) = \int_0^t Q(\tau) d\tau + V(0). \quad (5)$$

**6** We now consider two elements that can store energy, called **energy storage elements**; an element that can dissipate energy to a system's environment, called an **energy dissipative element**; and two elements that can supply power from outside a system, called **source elements**.

### Fluid inertances

**7** When fluid flows through a pipe, it has a momentum associated with it. The more mass (fluid density by its volume) moving in one direction and the faster it moves, the more momentum. This is stored kinetic energy. The discrete element we now introduce models this aspect of fluid systems.

**8** A **fluid inertance** is defined as an element for which the pressure momentum  $\Gamma$  across it is a monotonic function of the volumetric flowrate  $Q$  through it. A **linear inertance** is such that

$$\Gamma(t) = IQ(t), \quad (6)$$

where  $I$  is called the **inertance** and is typically a function of pipe geometry and fluid properties. This is called the element's **constitutive equation** because it constitutes what it means to be an inertance.

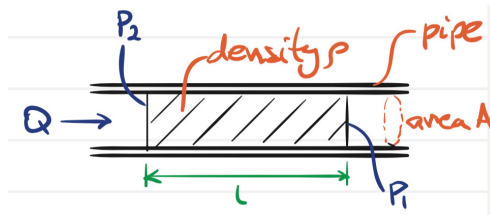
**9** Although there are nonlinear inertances, we can often use a linear model for analysis in some operating regime. The **elemental equation** for a linear inertance can be found by time-differentiating Equation 6 to obtain

We call this the elemental equation because it relates the element's power-flow variables  $Q$  and  $P$ .

**10** An inertance stores energy as kinetic energy, making it an *energy storage element*. The amount of energy it stores depends on the volumetric flowrate

it contains. For a linear inertance,

$$\varepsilon(t) = \frac{1}{2}IQ(t)^2. \quad (7)$$



**Figure flu.1:** a section of pipe for deriving its inertance.

**11** The inertance  $I$  for a uniform pipe can be derived, as follows, with reference to the sectioned pipe of Fig. flu.1. For an incompressible fluid flowing through a pipe of uniform area  $A$  and length  $L$ , with uniform velocity profile (a convenient fiction), an element of fluid obey's Newton's second law, from which several interesting equalities can be derived:

$$\begin{aligned} F &= m \frac{dv}{dt} \Rightarrow \\ \frac{F}{A} &= \frac{m}{A} \frac{dv}{dt} \Rightarrow \\ P &= \frac{\rho AL}{A} \frac{dv}{dt} \\ &= \rho L \frac{d}{dt} \left( \frac{Q}{A} \right) \\ &= \frac{\rho L}{A} \frac{dQ}{dt} \Rightarrow \\ \frac{dQ}{dt} &= \underbrace{\frac{A}{\rho L}}_{1/I} P. \end{aligned}$$

**12** From this last equality, it is clear that, for a uniform pipe and the assumptions, above,

$$I = \frac{\rho L}{A}. \quad (8)$$

Clearly, *long, thin* pipes will have more inertance. In fact, we often ignore inertance in modeling a pipe, unless it is relatively long and thin.

### Fluid capacitors

**13** When fluid is stored in tanks or in pressure vessels, it stores potential energy via its pressure drop  $P$ . For instance, a tank with a column of fluid will have a pressure drop associated with the height of the column. This is analogous to how an electronic capacitor stores its energy via its voltage. For this reason, we call such fluid elements **fluid capacitors**.

**14** A linear fluid capacitor with capacitance  $C$ , pressure drop  $P$ , and volume  $V$  has the constitutive equation

$$V = CP. \quad (9)$$

Once again, time-differentiating the constitutive equation gives us the elemental equation:



**15** Fluid capacitors can store energy (making them *energy storage elements*) in fluid potential energy, which, for a linear capacitor is

$$\mathcal{E}(t) = \frac{1}{2}CP^2. \quad (10)$$

### Fluid resistors

**Fluid resistors** are defined as elements for which the volumetric flowrate  $Q$  through the element is a monotonic function of the pressure drop  $P$  across it.

**Linear fluid resistors** have constitutive equation (and, it turns out, elemental equation)

$$Q = \frac{1}{R}P \quad (11)$$

where  $R$  is called the **fluid resistance**.

**16** Fluid resistors dissipate energy from the system (to heat), making them *energy dissipative elements*.

### Flowrate and pressure drop sources

**17** Fluid sources include pumps, runoff, etc.

**18** An **ideal volumetric flowrate source** is an element that provides arbitrary energy to a system via an independent (of the system) volumetric flowrate. The corresponding pressure drop across the element depends on the system.

**19** An **ideal pressure drop source** is an element that provides arbitrary energy to a system via an independent (of the system) pressure drop. The corresponding volumetric flowrate through the element depends on the system.

**20** Real sources, usually pumps, cannot be ideal sources, but in some instances can approximate them. More typical is to include a fluid resistor in tandem with an ideal source, as we did with electrical resistors for real electrical sources.

### Generalized element and variable types

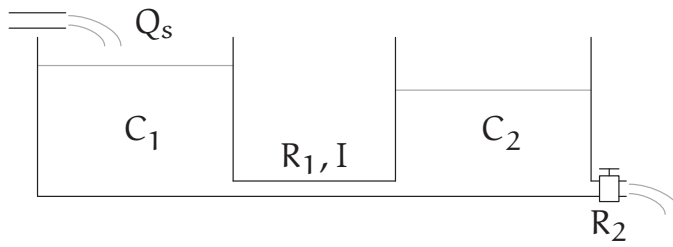
**21** In keeping with the definitions of [Chapter 01 intro](#), pressure  $P$  is an **across-variable** and flowrate  $Q$  is a **through-variable**.

**22** Consequently, the fluid capacitor is considered an **A-type** energy storage element. Similarly, the fluid inertance is a **T-type** energy storage element. Clearly, a fluid resistor is a **D-type** energy dissipative element.

**23** Pressure sources are, then, across-variable sources and volumetric flowrate sources are through-variable sources.

**Example 08.1 thermoflu.flu-1**

Use the schematic in Fig. flu.2 to draw a linear graph of the system.



**Figure flu.2:** schematic of a fluid system for Example 08.3 thermoflu.flutrans-1.

re:  
fluid  
system  
graph

## 08.2 thermoflu.therm Thermal system elements

- 1 Systems in which heat flow is of interest are called **thermal systems**. For instance, heat generated by an engine or a server farm flows through several bodies via the three modes of heat transfer: **conduction**, **convection**, and **radiation**. This is, of course, a dynamic process.
- 2 A detailed model would require a spatial continuum. However, we are often concerned with, say, the maximum temperature an engine will reach for different speeds or the maximum density of a server farm while avoiding overheating. Or, more precisely, *how* a given heat generation affects the temperature response of system components.
- 3 As with electrical, mechanical, and fluid systems, we can describe thermal systems as consisting of discrete lumped-parameter elements. The dynamic models that can be developed from considering these elements are often precisely the right granularity for system-level design.
- 4 We now introduce a few lumped-parameter elements for modeling thermal systems. Let a **heat flow rate**  $q$  (SI units W) and **temperature**  $T$  (SI units K or C) be input to a port in a thermal element. There are three structural differences between thermal systems and the other types we've considered. We are confronted with the **first**, here, when we consider that heat power is typically *not* considered to be the product of two variables; rather, the heat flow rate  $q$  is *already power*:

$$\mathcal{P}(t) = q(t). \quad (1)$$

A thermal element has two distinct locations between which its temperature drop is defined. We call a reference temperature **ground**.

- 5 The **heat energy**  $H$  of a system with initial heat  $H(0)$  is

$$H(t) = \int_0^t \mathcal{P}(\tau) d\tau + H(0). \quad (2)$$

- 6 We now consider an element that can store energy, called an **energy storage element**; an element that resists power flow; and two elements that can supply power from outside a system, called **source elements**. The



**second** difference is that there is only one type of energy storage element in the thermal domain.

### Thermal capacitors

7 When heat is stored in an object, it stores potential energy via its temperature  $T$ . This is analogous to how an electronic capacitor stores its energy via its voltage. For this reason, we call such thermal elements **thermal capacitors**.

8 A linear thermal capacitor with thermal capacitance  $C$  (SI units J/K), temperature  $T$ , and heat  $H$  has the constitutive equation

$$H = CT. \quad (3)$$

Once again, time-differentiating the constitutive equation gives us the elemental equation:



9 The thermal capacitance  $C$  is an **extensive property**—that is, it depends on the amount of its substance. This is opposed to the **specific heat capacity**  $c$  (units J/K/kg), an **intensive property**: one that is independent of the amount of its substance. These quantities are related for an object of mass  $m$  by the equation

$$C = mc. \quad (4)$$

### Thermal resistors

10 **Thermal resistors** are defined as elements for which the heat flowrate  $q$  through the element is a monotonic function of the temperature drop  $T$  across it. **Linear thermal resistors** have constitutive equation (and, it turns out, elemental equation)

$$q = \frac{1}{R}T \quad (5)$$

where  $R$  is called the **thermal resistance**.

**11** Thermal resistors do *not* dissipate energy from the system, which is the **third** difference between thermal and other energy domains we've considered. After all, the other "resistive" elements all dissipate energy *to heat*. Rather than dissipate energy, they simply impede its flow.

**12** All three modes of heat transfer are modeled by thermal resistors, but only two of them are well-approximated as linear for a significant range of temperature.

**conduction** Heat conduction is the transfer of heat through an object's microscopic particle interaction.<sup>1</sup> It is characterized by a thermal resistance

$$R = \frac{L}{\rho A}, \quad (6)$$

where  $L$  is the length of the object *in the direction of heat transfer*,  $A$  is the transverse cross-sectional area, and  $\rho$  is the material's **thermal conductivity** (SI units  $W/K/m$ ).<sup>2</sup>

**convection** Heat convection is the transfer of heat via **fluid advection**: the bulk motion of a fluid. It is characterized by a thermal resistance

$$R = \frac{1}{hA}, \quad (7)$$

where  $h$  is the **convection heat transfer coefficient** (SI units  $W/m^2/K$ ) and  $A$  is the area of fluid-object contact (SI units  $m^2$ ). The convection heat transfer coefficient  $h$  is highly and nonlinearly dependent on the velocity of the fluid. Furthermore, the geometry of the objects and the fluid composition affect  $h$ .

**radiation** Radiative heat transfer is electromagnetic radiation emitted from one body and absorbed by another. For  $T_1$  the absolute temperature of a "hot" body,  $T_2$  the absolute temperature of a "cold"

<sup>1</sup>We use the term "object" loosely, here, to mean a grouping of continuous matter in any phase.

<sup>2</sup>Thermal resistance can also be defined as an intensive property  $\rho^{-1}$ , the reciprocal of the thermal conductivity. Due to our lumped-parameter perspective, we choose the extensive definition.

body,  $\epsilon$  the **effective emissivity/absorptivity**,<sup>3</sup> and  $A$  the area of the exposed surfaces, the heat transfer is characterized by

$$q = \epsilon \sigma A (T_1^4 - T_2^4), \quad (8)$$

where  $\sigma$  is the **Stefan-Boltzmann constant**

$$\sigma = 5.67 \cdot 10^{-8} \frac{\text{W}}{\text{m}^2\text{K}^4}. \quad (9)$$

Clearly, this heat transfer is highly nonlinear. Linearization of this heat transfer is problematic because the temperature difference  $T$  between the bodies does not appear in the expression. For many system models, radiative heat transfer is assumed negligible. We must be cautious with this assumption, however, especially when high operating temperatures are anticipated.

### Heat flow rate and temperature sources

**13** Thermal sources include many physical processes—almost everything generates heat!

**14** An **ideal heat flow rate source** is an element that provides arbitrary heat flow rate  $Q_s$  to a system, independent of the temperature across it, which depends on the system.

**15** An **ideal temperature source** is an element that provides arbitrary temperature  $T_s$  to a system, independent of the heat flow rate through it, which depends on the system.

### Generalized element and variable types

**16** In keeping with the definitions of [Chapter 01 intro](#), temperature  $T$  is an **across-variable** and heat flow rate  $q$  is a **through-variable**.

**17** Consequently, the thermal capacitor is considered an **A-type** energy storage element. A thermal resistor is considered to be a **D-type** energy dissipative element, although it does not actually dissipate energy. It does,

<sup>3</sup>The parameter  $\epsilon$  is taken to be the combined “gray body” emissivity/absorptivity. Consult a heat transfer text for details.

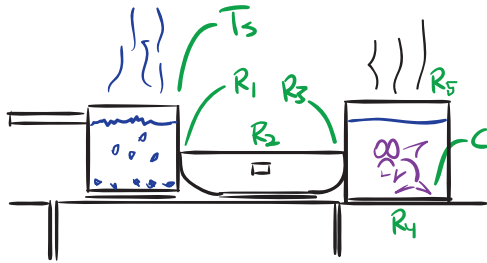
however, *resist* its flow and relates its across- and through-variables *algebraically*, both signature characteristics of D-type elements.

**18** Temperature sources are, then, across-variable sources and heat flow rate sources are through-variable sources.

### Example 08.2 thermoflu.therm-1

Careless Carlton left a large pot of water boiling on the stove. Worse, a cast-iron pan is bumped so that it is in solid contact with the pot *and* his glass fish tank, which was carelessly left next to the stove, as shown in Fig. therm.1. Draw a linear graph of the sad situation to determine what considerations determine if Careless Carlton's fish goes from winner to dinner.

re:  
thermal  
system  
graph



**Figure therm.1:** Careless Carlton's fish's sad situation.

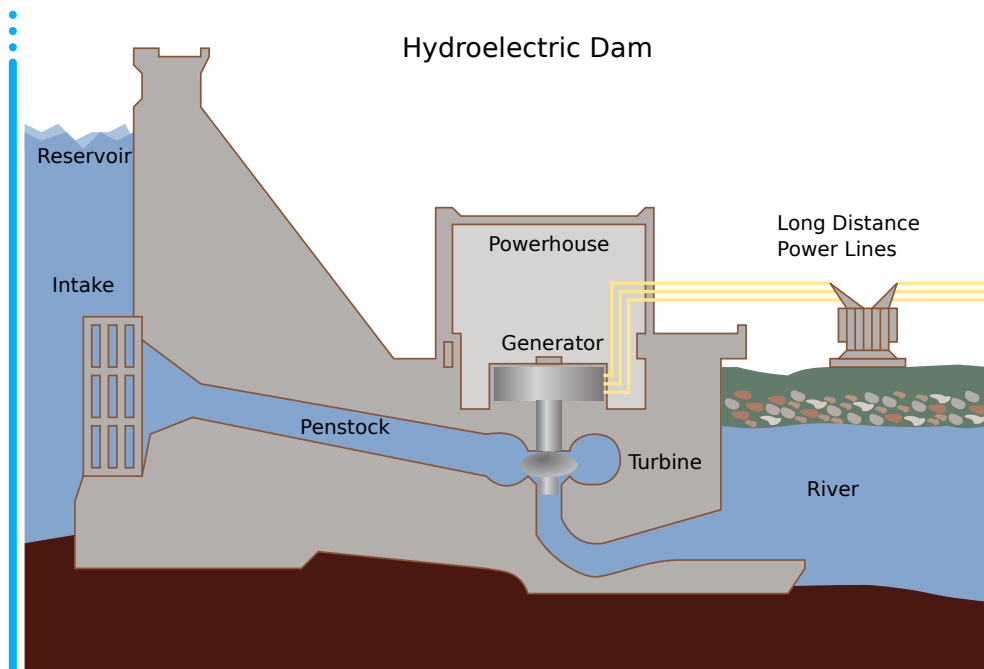
## 08.3 thermoflu.flutrans Fluid transducers

- 1 Although thermal systems often exchange energy with other energy domains, it is much more common to consider those systems that interact with thermal systems to be generating or sinking heat (often modeled as a *dependent source*) than to see a proper transducer.
- 2 Fluid systems, on the other hand, very naturally interact with mechanical systems. For instance, piston-cylinder mechanisms, propellers, turbines, and impellers (backward turbines) are just a few energy transducing elements.
- 3 These systems are often driven by motors (e.g. a pump's impeller) or drive generators (e.g. a dam's turbine). Therefore, it is common to require a fluid-electromechanical dynamic model.

### Example 08.3 thermoflu.flutrans-1

Dams, even small, “micro” dams, generate hydroelectric power by directing water through turbines, which rotate, creating mechanical power, and drive electric generators, generating electric power. For large-scale dams, the flowrate is regulated such that an AC generator produces a nice 60 Hz. However, a microhydroelectric generator typically cannot expect well-regulated flowrates, so sometimes they use a brushed DC generator (brush replacement being the primary drawback). Assuming a microhydroelectric dam can be set up in a manner similar to a large-scale dam, draw a linear graph model from the schematic of Fig. flutrans.1.

re:  
microhydroelectric  
power  
generation



**Figure flutrans.1:** schematic of a hydroelectric dam (Authority and Tomia, 2018).

## 08.4 thermoflu.dam State-space model of a hydroelectric dam

1 Consider the microhydroelectric dam of [Example 08.3 thermoflu.flutrans-1](#). We derived the linear graph of [Fig. dam.1](#). In this lecture, we will derive a state-space model for the system—specifically, a state equation.

### Normal tree, order, and variables

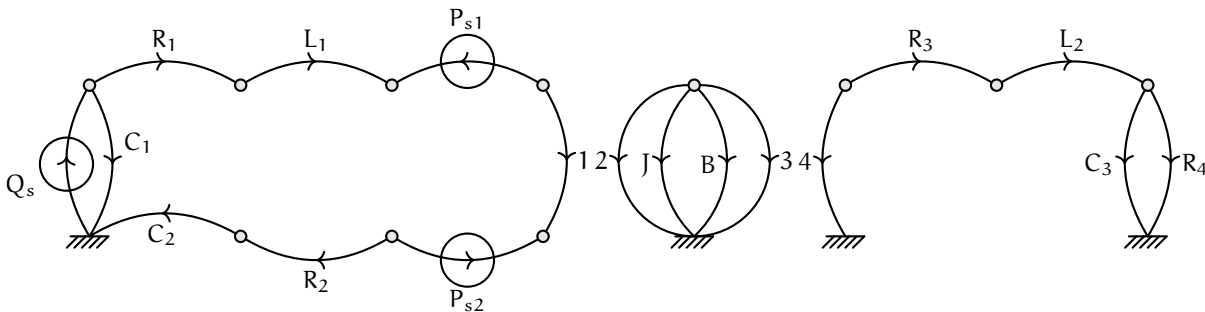
2 Now, we define a **normal tree** by overlaying it on the system graph in [Fig. dam.1](#). There are six independent energy storage elements, making it a sixth-order ( $n = 6$ ) system. We define the state vector to be

$$\mathbf{x} = [P_{C_1} \quad P_{C_2} \quad Q_{L_1} \quad \Omega_J \quad i_{L_2} \quad v_{C_3}]^T. \quad (1)$$

The input vector is defined as  $\mathbf{u} = [Q_s \quad P_{s1} \quad P_{s2}]^T$ .

### Elemental equations

3 Yet to be encountered is a turbine's transduction. A simple model is that the torque  $T_2$  is proportional to the flowrate  $Q_1$ , which are both



**Figure dam.1:** a linear graph for a microhydroelectric dam.

through-variables, making it a **transformer**, so

$$T_2 = -\alpha Q_1 \quad \text{and} \quad \Omega_2 = \frac{1}{\alpha} P_1, \quad (2)$$

where  $\alpha$  is the **transformer ratio**.

4 The other elemental equations have been previously encountered and are listed, below.

el.	elemental eq.	el.	elemental eq.	el.	elemental eq.
$C_1$	$\frac{dP_{C_1}}{dt} = \frac{1}{C_1} Q_{C_1}$	$C_3$	$\frac{dv_{C_3}}{dt} = \frac{1}{C_3} i_{C_3}$	B	$\Omega_B = \frac{1}{B} T_B$
$C_2$	$\frac{dP_{C_2}}{dt} = \frac{1}{C_2} Q_{C_2}$	$R_1$	$P_{R_1} = Q_{R_1} R_1$	3	$i_4 = \frac{-1}{k_m} T_3$
$L_1$	$\frac{dQ_{L_1}}{dt} = \frac{1}{L_1} P_{L_1}$	$R_2$	$P_{R_2} = Q_{R_2} R_2$	4	$v_4 = k_m \Omega_3$
J	$\frac{d\Omega_J}{dt} = \frac{1}{J} T_J$	1	$T_2 = -\alpha Q_1$	$R_3$	$v_{R_3} = i_{R_3} R_3$
$L_2$	$\frac{di_{L_2}}{dt} = \frac{1}{L_2} v_{L_2}$	2	$\Omega_2 = \frac{1}{\alpha} P_1$	$R_4$	$i_{R_4} = \frac{1}{R_4} v_{R_4}$

### Continuity and compatibility equations

5 Continuity and compatibility equations can be found in the usual way—by drawing contours and temporarily creating loops by including links in the normal tree. We proceed by drawing a table of all elements and writing a continuity equation for each branch of the normal tree and a compatibility equation for each link.



el.	eq.	el.	eq.	el.	eq.
C <sub>1</sub>	$Q_{C_1} = Q_s - Q_{L_1}$	C <sub>3</sub>	$i_{C_3} = i_{L_2} - i_{R_4}$	B	$\Omega_B = \Omega_J$
C <sub>2</sub>	$Q_{C_2} = Q_{L_1}$	R <sub>1</sub>	$Q_{R_1} = Q_{L_1}$	3	$\Omega_3 = \Omega_J$
L <sub>1</sub>	$P_{L_1} = -P_{R_1} + P_{C_1} - P_{C_2} +$ $-P_{R_2} + P_{s2} - P_1 + P_{s1}$	R <sub>2</sub>	$Q_{R_2} = Q_{L_1}$	4	$i_4 = -i_{L_2}$
J	$T_J = -T_2 - T_B - T_3$	1	$Q_1 = Q_{L_1}$	R <sub>3</sub>	$i_{R_3} = i_{L_2}$
L <sub>2</sub>	$v_{L_2} = -v_{R_3} + v_4 - v_{C_3}$	2	$\Omega_2 = \Omega_J$	R <sub>4</sub>	$v_{R_4} = v_{C_3}$

### State equation

6 The system of equations composed of the elemental, continuity, and compatibility equations can be reduced to the state equation. There is a substantial amount of algebra required to eliminate those variables that are neither state nor input variables. Therefore, we use the Mathematica package *StateMint* (Devine and Picone, 2018). The resulting system model is:

$$\frac{dx}{dt} = Ax + Bu,$$

$$A = \begin{bmatrix} 0 & 0 & -1/C_1 & 0 & 0 & 0 \\ 0 & 0 & 1/C_2 & 0 & 0 & 0 \\ 1/L_1 & -1/L_1 & -(R_1 + R_2)/L_1 & -\alpha/L_1 & 0 & 0 \\ 0 & 0 & \alpha/J & -B/J & -k_m/J & 0 \\ 0 & 0 & 0 & k_m/L_2 & -R_3/L_2 & -1/L_2 \\ 0 & 0 & 0 & 0 & 1/C_3 & -1/(R_4 C_3) \end{bmatrix},$$

$$B = \begin{bmatrix} 1/C_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/L_1 & 1/L_1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

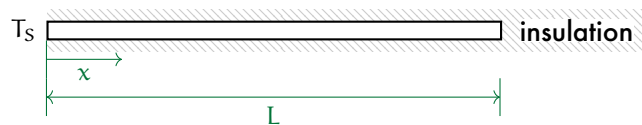
7 The rub is estimating all these parameters.

8 The Mathematica notebook used above can be found in the [source repository](#) for this text.

## 08.5 thermoflu.fem Thermal finite element model

### Example 08.5 thermoflu.fem-1

re:  
thermal  
finite  
element  
model



**Figure fem.1:** an insulated bar.

Consider the long homogeneous copper bar of Fig. fem.1, insulated around its circumference, and initially at uniform temperature. At time  $t = 0$ , the temperature at one end of the bar ( $x = 0$ ) is increased by one Kelvin. We wish to find the dynamic variation of the temperature at any location  $x$  along the bar, at any time  $t > 0$ .

Construct a discrete element model of thermal conduction in the bar, for which the following parameters are given for its length  $L$  and diameter  $d$ .

```
L = 1; % m
d = 0.01; % m
```

#### Geometrical considerations

The cross-sectional area for the bar is as follows.

```
a = pi/4*d^2; % m^2 x-sectional area
```

- Dividing the bar into  $n$  sections ("finite elements") such that we have length
- of each  $dx$  gives the following.

```
n = 100; % number of chunks
dx = L/n; % m ... length of chunk
```

### Material considerations

The following are the material properties of copper.

```
cp = 390; % SI ... specific heat capacity
rho = 8920; % SI ... density
ks = 401; % SI ... thermal conductivity
```

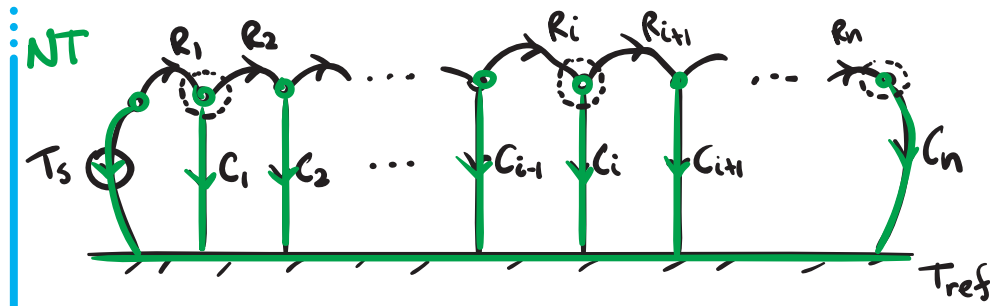
### Lumping

From the geometrical and material considerations above, we can develop a lumped thermal resistance  $R$  and thermal capacitance  $c$  of each cylindrical section of the bar of length  $dx$ . From Eq. 6 and Eq. 4, these parameters are as follows.

```
R = dx/(ks*a); % thermal resistance
dV = dx*a; % m^3 ... section volume
dm = rho*dV; % kg ... section mass
c = dm*cp; % section volume
```

### Linear graph model

- The linear graph model is shown in Fig. fem.2 with the corresponding
- normal tree overlaid.



**Figure fem.2:** a linear graph of the insulated bar.

*State-space model*

The state variables are clearly the temperatures of  $C_i$ :  $T_{C_1}, \dots, T_{C_n}$ . Therefore, the order of the system is  $n$ .

The state, input, and output variables are

$$\mathbf{x} = [T_{C_1} \dots T_{C_n}]^T, \quad \mathbf{u} = [T_S], \quad \text{and} \quad \mathbf{y} = \mathbf{x}. \quad (1)$$

**Elemental, continuity, and compatibility equations** Consider the elemental, continuity, and compatibility equations, below, for the first, a middle, and the last elements. The following makes the assumption of homogeneity, which yields  $R_i = R$  and  $C_i = C$ .

element	elemental eq.	continuity eq.	compatibility eq.
$C_1$	$\dot{T}_{C_1} = \frac{1}{C} q_{C_1}$	$q_{C_1} = q_{R_1} - q_{R_2}$	
$R_1$	$q_{R_1} = \frac{1}{R} T_{R_1}$		$T_{R_1} = T_S - T_{C_1}$
$C_i$	$\dot{T}_{C_i} = \frac{1}{C} q_{C_i}$	$q_{C_i} = q_{R_i} - q_{R_{i+1}}$	
$R_i$	$q_{R_i} = \frac{1}{R} T_{R_i}$		$T_{R_i} = T_{C_{i-1}} - T_{C_i}$
$C_n$	$\dot{T}_{C_n} = \frac{1}{C} q_{C_n}$	$q_{C_n} = q_{R_n}$	
$R_n$	$q_{R_n} = \frac{1}{R} T_{R_n}$		$T_{R_n} = T_{C_{n-1}} - T_{C_n}$

- **Deriving the state equations for sections 1, i, and n** For each of the first, a representative middle, and the last elements, we can derive the state

⋮ equation with relatively few substitutions, as follows.

$$\begin{aligned}
 \dot{T}_{C_1} &= \frac{1}{C} q_{C_1} && \text{(elemental)} \\
 &= \frac{1}{C} (q_{R_1} - q_{R_2}) && \text{(continuity)} \\
 &= \frac{1}{RC} (T_{R_1} - T_{R_2}) && \text{(elemental)} \\
 &= \frac{1}{RC} (T_S - T_{C_1} - T_{C_1} + T_{C_2}) && \text{(compatibility)} \\
 &= \frac{1}{RC} (T_S - 2T_{C_1} + T_{C_2}). \\
 \dot{T}_{C_i} &= \frac{1}{C} q_{C_i} && \text{(elemental)} \\
 &= \frac{1}{C} (q_{R_i} - q_{R_{i+1}}) && \text{(continuity)} \\
 &= \frac{1}{RC} (T_{R_i} - T_{R_{i+1}}) && \text{(elemental)} \\
 &= \frac{1}{RC} (T_{C_{i-1}} - 2T_{C_i} + T_{C_{i+1}}). && \text{(compatibility)} \\
 \dot{T}_{C_n} &= \frac{1}{C} q_{C_n} && \text{(elemental)} \\
 &= \frac{1}{C} q_{R_n} && \text{(continuity)} \\
 &= \frac{1}{RC} T_{R_n} && \text{(elemental)} \\
 &= \frac{1}{RC} (T_{C_{n-1}} - T_{C_n}). && \text{(compatibility)}
 \end{aligned}$$

- Let  $\tau = RC$ . The A and B matrices are, then

$$A = \begin{bmatrix} -2/\tau & 1/\tau & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1/\tau & -2/\tau & 1/\tau & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ & & & \ddots & \ddots & \ddots & & & & & \\ \vdots & & & & 1/\tau & -2/\tau & 1/\tau & & & & \vdots \\ & & & & & \ddots & \ddots & \ddots & & & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1/\tau & -2/\tau & 1/\tau \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1/\tau & -1/\tau \end{bmatrix}$$

$$B = \begin{bmatrix} 1/\tau \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n \times 1} \quad (2)$$

The outputs are the states:  $\mathbf{y} = \mathbf{x}$ . Or, in standard form with identity matrix I, the matrices are:

$$C = I_{n \times n} \quad \text{and} \quad D = 0_{n \times 1}. \quad (3)$$

*Simulation of a step response*

Define the A matrix.

```
A = zeros(n);
% first row
A(1,1) = -2/(R*c);
A(1,2) = 1/(R*c);
% last row
A(n,n-1) = 1/(R*c);
A(n,n) = -1/(R*c);
% middle rows
for i = 2:(n-1)
    A(i,i-1) = 1/(R*c);
    A(i,i) = -2/(R*c);
    A(i,i+1) = 1/(R*c);
end
```

- Now define B, C, and D.

```
B = zeros([n,1]);  
B(1) = 1/(R*c);  
C = eye(n);  
D = zeros([n,1]);
```

Create a state-space model.

```
sys = ss(A,B,C,D);
```

Simulate a unit step in the input temperature.

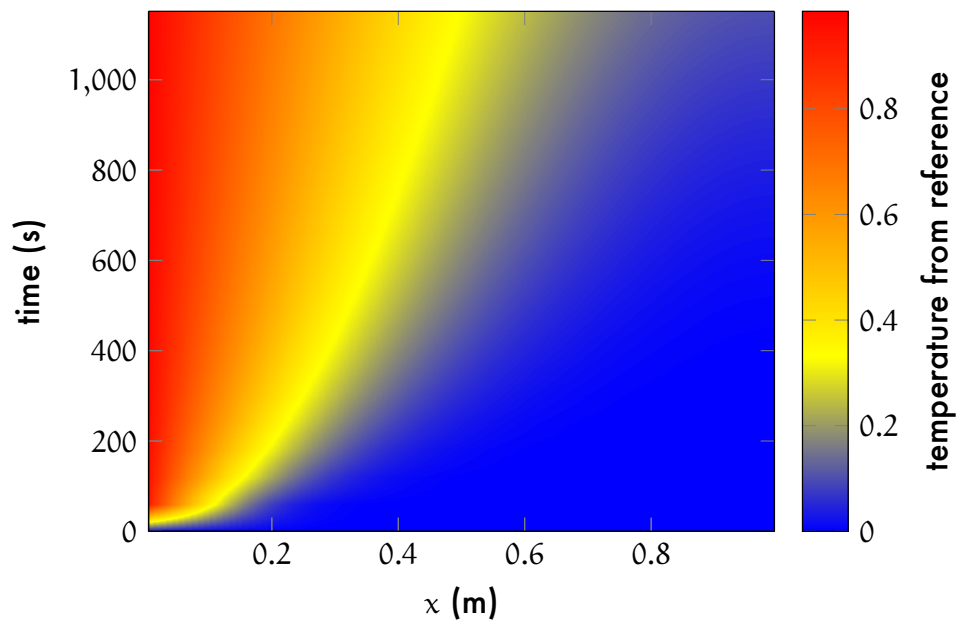
```
Tmax = 1200; % sec ... final sim time  
t = linspace(0,Tmax,100);  
y = step(sys,t);
```

**Plot the step response** To prepare for creating a 3D plot, we need to make a grid of points.

```
x = dx/2:dx:(L-dx/2);  
[X,T] = meshgrid(x,t);
```

• Now we're ready to plot. The result is shown in Fig. fem.3.





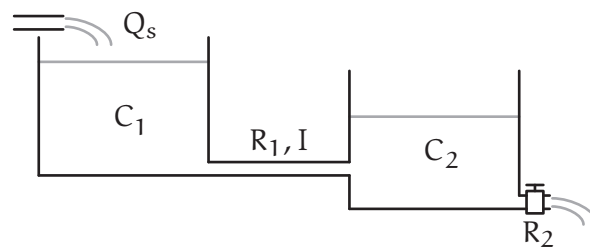
**Figure fem.3:** spatiotemporal thermal response.

```
figure
contourf(X,T,y)
shading(gca,'interp')
xlabel('x')
ylabel('time')
zlabel('temp (K)')
```

## 08.6 thermoflu.exe Exercises for Chapter 08 thermoflu

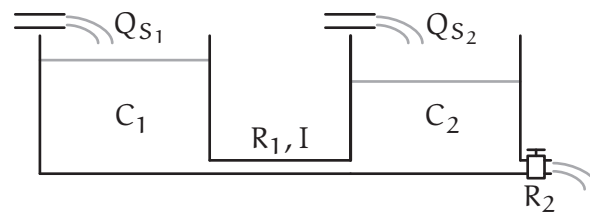
### Exercise 08.1 tinker

Draw a linear graph of the fluid system with schematic below.



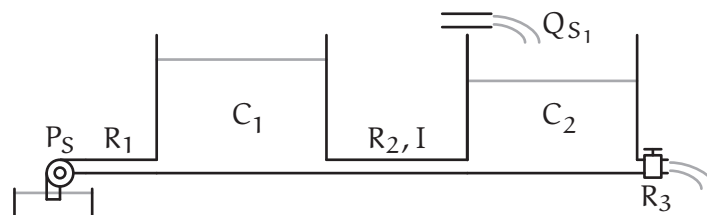
### Exercise 08.2 tailor

Draw a linear graph of the fluid system with schematic below.



### Exercise 08.3 soldier

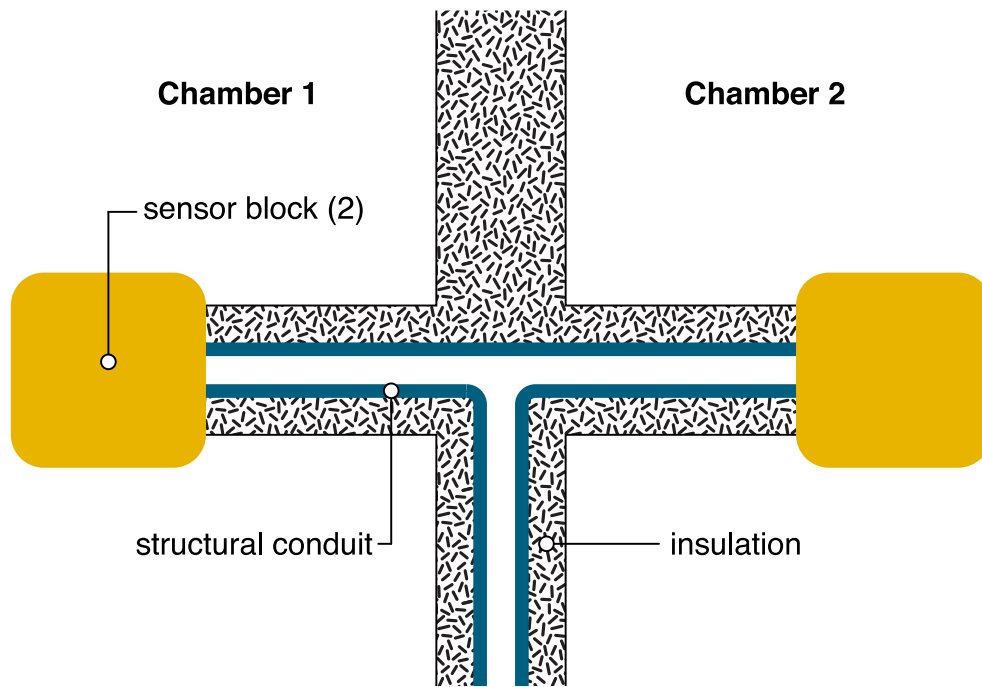
(a) Draw a linear graph of the fluid system with schematic below. (b) Draw a normal tree and identify the state variables and system order.



**Exercise 08.4 tpain**

Consider an apparatus with two chambers filled with gas at potentially different temperatures illustrated in Fig. exe.1. Temperature sensors are embedded in the two “sensor blocks,” made of copper for low thermal resistance and made large enough to provide enough thermal capacitance to smooth out temperature fluctuations.<sup>4</sup> The “structural conduit” is made of steel, less thermally conductive, but conductive nonetheless. The conduit provides structure to the apparatus and is hollow to allow the sensor wires to run through.

\_\_\_\_\_/  
25 p.



**Figure exe.1:** a diagram of the two-chamber apparatus.

A concern with this apparatus is that the temperature in one chamber will affect the temperature in the other, most conspicuously by heat conducting through the structural conduit.

<sup>4</sup>This technique of adding capacitance for smoothing a signal is useful in all energy domains!

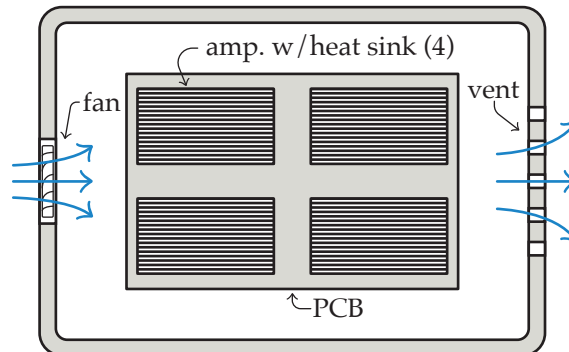
We will begin an analysis of the thermal isolation of the two chambers and temperature measurements. Develop a thermal lumped-parameter model as follows.

- Describe the lumped-parameter elements you will use to model the system.
- Draw a linear graph of the lumped-parameter model.
- Superimpose a normal tree on the graph, identify the system order, and choose the state variables.

### Exercise 08.5 dramp

Consider a device with four amplifiers in an array on a printed circuit board (PCB), as illustrated in Fig. exe.2. The amplifiers generate significant heat (as a byproduct), and they must be cooled. For this reason, each amplifier has mounted on top a heat sink device with fins. A fan forces airflow over the fins to dissipate the heat via convection.

\_\_\_\_\_/25 p.



**Figure exe.2:** Top view of four amplifiers in a chassis.

As the designer of the chassis housing the amplifiers, you would like to develop a lumped-parameter thermal model of the system to ensure that, under different heat generation loads, the amplifiers remain within their acceptable temperature range.

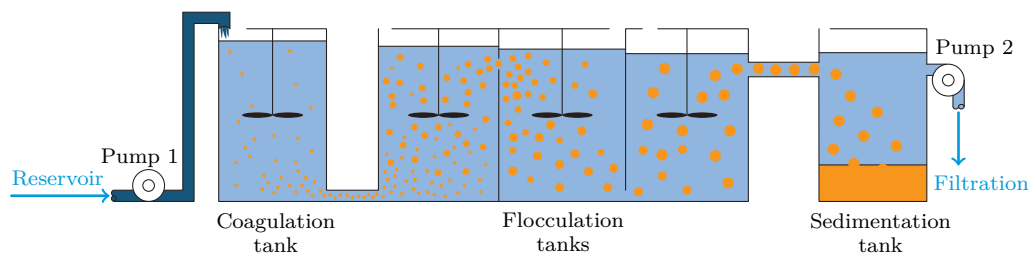
- Describe the lumped-parameter elements you will use to model the system, including inputs.

- b. Draw a linear graph of the lumped-parameter model.
- c. Superimpose a normal tree on the graph, identify the system order, and choose the state variables.

### Exercise 08.6 up

Consider the diagram of the first stages of a drinking water treatment plant shown in Fig. exe.3. The water to be treated comes from a reservoir and is pumped by Pump 1 into the coagulation tank. The suspended particles are too small to settle via gravity, and their generally negative charges repel each other, keeping them from clumping. Here a small amount of a chemical coagulant with positive charge is rapidly mixed in with paddles. Coagulation is the resultant clumping of the particles.

\_\_\_\_\_/ 35 p.



**Figure exe.3:** A water treatment plant.

The water with coagulated particles flows through a long, thin pipe and enters a series of 3 flocculation tanks. In the flocculation tanks, which mix at decreasing rates as the fluid progresses, the coagulated particles join into increasingly larger pieces called flocs. Note that the placement of the inlet and outlet of each tank has a sorting effect.

After the third tank, the water flows through a short, wide pipe into the sedimentation tank. Now the flocs are large enough to settle via gravity, and Pump 2 on the outlet pumps the water sans flocs to filtration and disinfection stages of the purification process.

The quality of the process is highly dependent on the volumetric flow rates and pressures in the tanks. Develop a lumped-parameter linear graph model with the following steps:

- a. Describe the lumped-parameter elements you will use to model the system, including inputs.
- b. Draw a linear graph of the lumped-parameter model.
- c. Superimpose a normal tree on the graph, identify the system order, and choose the state variables.

## **Part IV**

# **Fourier analysis**

**09 four**

---



## 09.1 four.series Fourier series

**1** Fourier series are mathematical series that can represent a periodic signal as a sum of sinusoids at different amplitudes and frequencies. They are useful for solving for the response of a system to periodic inputs. However, they are probably most important *conceptually*: they are our gateway to thinking of signals in the **frequency domain**—that is, as functions of *frequency* (not time). To represent a function as a Fourier series is to **analyze** it as a sum of sinusoids at different frequencies<sup>1</sup>  $\omega_n$  and amplitudes  $a_n$ . Its **frequency spectrum** is the functional representation of amplitudes  $a_n$  versus frequency  $\omega_n$ .

**2** Let's begin with the definition.

### Definition 09 four.1: Fourier series: trigonometric form

The *Fourier analysis* of a periodic function  $y(t)$  is, for  $n \in \mathbb{N}_0$ , period  $T$ , and angular frequency  $\omega_n = 2\pi n/T$ ,

$$a_0 = \frac{2}{T} \int_{-T/2}^{T/2} y(t) dt \quad (1)$$

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} y(t) \cos(\omega_n t) dt \quad (2)$$

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} y(t) \sin(\omega_n t) dt. \quad (3)$$

The *Fourier synthesis* of a periodic function  $y(t)$  with analysis components  $a_n$  and  $b_n$  corresponding to  $\omega_n$  is

$$y(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(\omega_n t) + b_n \sin(\omega_n t). \quad (4)$$

**3** Let's consider the complex form of the Fourier series, which is analogous to [Definition 09 four.1](#). It may be helpful to review Euler's formula(s) – see [Appendix 04.1 com.euler](#).

<sup>1</sup>It's important to note that the symbol  $\omega_n$ , in this context, is not the natural frequency, but a frequency indexed by integer  $n$ .

**Definition 09 four.2: Fourier series: complex form**

The *Fourier analysis* of a periodic function  $y(t)$  is, for  $n \in \mathbb{N}_0$ , period  $T$ , and angular frequency  $\omega_n = 2\pi n/T$ ,

$$c_{\pm n} = \frac{1}{T} \int_{-T/2}^{T/2} y(t) e^{-j\omega_n t} dt. \quad (5)$$

The *Fourier synthesis* of a periodic function  $y(t)$  with analysis components  $c_n$  corresponding to  $\omega_n$  is

$$y(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega_n t}. \quad (6)$$

4 We call the integer  $n$  a **harmonic** and the frequency associated with it,

$$\omega_n = 2\pi n/T, \quad (7)$$

the **harmonic frequency**. There is a special name for the first harmonic ( $n = 1$ ): the **fundamental frequency**. It is called this because all other frequency components are integer multiples of it.

5 It is also possible to convert between the two representations above.

**Definition 09 four.3: Fourier series: converting between forms**

The complex Fourier analysis of a periodic function  $y(t)$  is, for  $n \in \mathbb{N}_0$  and  $a_n$  and  $b_n$  as defined above,

$$c_{\pm n} = \frac{1}{2} (a_{|n|} \mp j b_{|n|}) \quad (8)$$

The sinusoidal Fourier analysis of a periodic function  $y(t)$  is, for  $n \in \mathbb{N}_0$  and  $c_n$  as defined above,

$$a_n = c_n + c_{-n} \text{ and} \quad (9)$$

$$b_n = j(c_n - c_{-n}). \quad (10)$$

6 The **harmonic amplitude**  $C_n$  is

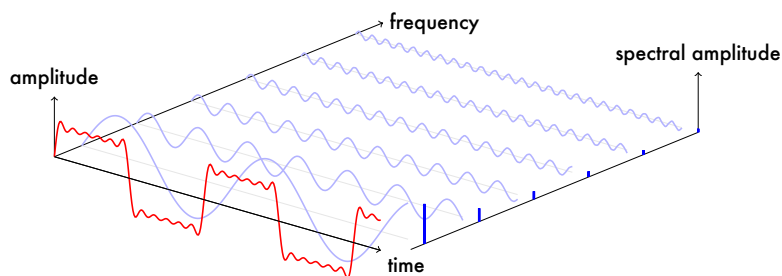
$$C_n = \sqrt{a_n^2 + b_n^2} \quad (11)$$

$$= 2\sqrt{c_n c_{-n}}. \quad (12)$$

A **magnitude line spectrum** is a graph of the harmonic amplitudes as a function of the harmonic frequencies. The **harmonic phase** is

$$\begin{aligned}\theta_n &= -\arctan_2(b_n, a_n) && \text{(see Appendix 01.2 math.trig)} \\ &= \arctan_2(\operatorname{Im}(c_n), \operatorname{Re}(c_n)). && (13)\end{aligned}$$

7 The illustration of Fig. series.1 shows how sinusoidal components sum to represent a square wave. A line spectrum is also shown.



**Figure series.1:** a partial sum of Fourier components of a square wave shown through time and frequency. The spectral amplitude shows the amplitude of the corresponding Fourier component.

8 Let us compute the associated spectral components in the following example.

### Example 09.1 four.series-1

Compute the first five harmonic amplitudes that represent the line spectrum for a square wave in the figure above.

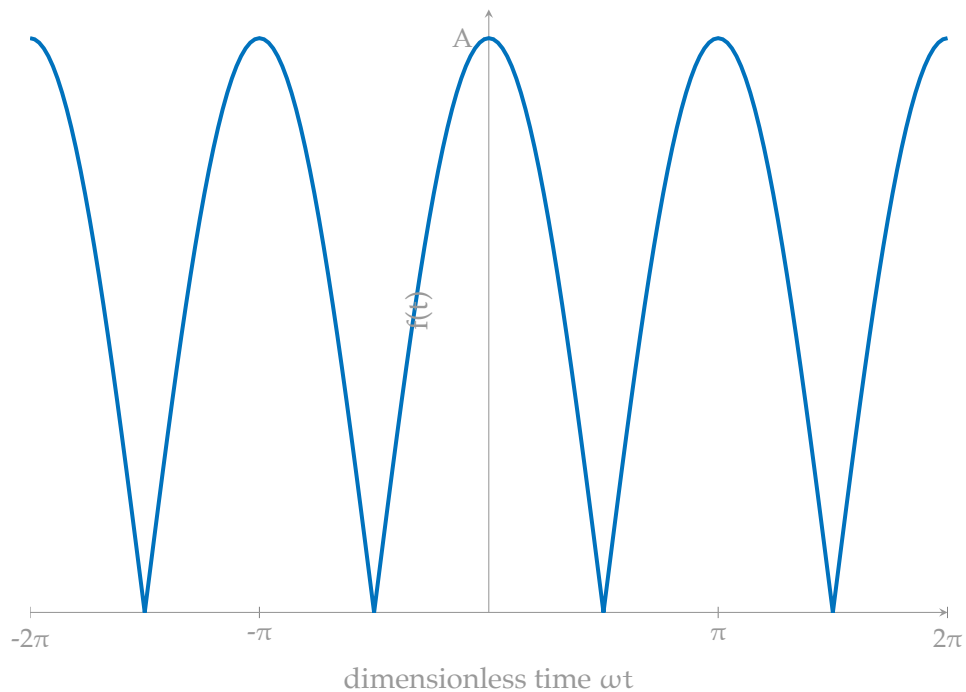
re:  
Fourier  
series  
analysis:  
line  
spectrum



## 09.2 four.fsex.a Complex Fourier series example

1 There are several flavors of Fourier series problem: trigonometric/exponential, analysis/synthesis, plotting partial sums/plotting spectra. Of course, problems just present us an opportunity to traverse part of the landscape (to mix two metaphors like 31 similies).

### Example 09.2 four.fsex.a-1



**Figure fsex.a.1:** the function  $f(t) = |A \cos(\omega t)|$  plotted for several periods.

2 Consider a rectified sinusoid

$$f(t) = |A \cos(\omega t)|$$

••• for  $A, \omega, t \in \mathbb{R}$ , shown in Fig. fsex.1. The fundamental period is  $T = \pi/\omega$ , half the unrectified period.

- a. Perform a complex Fourier analysis on  $f(t)$ , computing the complex Fourier components  $c_{\pm n}$ .
- b. Compute and plot the magnitude and phase spectra.
- c. Convert  $c_{\pm n}$  to trigonometric components  $a_n$  and  $b_n$ .

### Part a: complex Fourier analysis

3 The complex Fourier analysis of Definition 09 four.2 will be applied in a moment. However, it is convenient to first convert  $f$  into an \_\_\_\_\_ . We can write  $f$  over a single period  $t \in [-T/2, T/2)$  as

(absolute value property)

(already positive)

(Euler, Eq. 2)

••• 4 Applying Fourier analysis *à la* Definition 09 four.2 with harmonic

• frequency  $\omega_n = 2\pi n/T$ ,

$$\begin{aligned}
 c_{\pm n} &= \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j\omega_n t} dt \\
 &= \frac{1}{T} \int_{-T/2}^{T/2} |A| \frac{1}{2} (e^{j\omega t} + e^{-j\omega t}) e^{-j\omega_n t} dt \\
 &= \frac{|A|}{2T} \int_{-T/2}^{T/2} (e^{j\omega t} + e^{-j\omega t}) e^{-j\omega_n t} dt \\
 &= \frac{|A|}{2T} \int_{-T/2}^{T/2} (e^{j(\omega - \omega_n)t} + e^{-j(\omega + \omega_n)t}) dt \\
 &= \frac{|A|}{2T} \left( \frac{1}{j(\omega - \omega_n)} e^{j(\omega - \omega_n)t} - \frac{1}{j(\omega + \omega_n)} e^{-j(\omega + \omega_n)t} \right) \Big|_{-T/2}^{T/2} \\
 &= \frac{|A|}{2T} \left( \frac{1}{j(\omega - \omega_n)} e^{j(\omega - \omega_n)T/2} - \frac{1}{j(\omega + \omega_n)} e^{-j(\omega + \omega_n)T/2} + \right. \\
 &\quad \left. - \frac{1}{j(\omega - \omega_n)} e^{-j(\omega - \omega_n)T/2} + \frac{1}{j(\omega + \omega_n)} e^{j(\omega + \omega_n)T/2} \right) \\
 &= \frac{|A|}{j2T(\omega - \omega_n)} (e^{j(\omega - \omega_n)T/2} - e^{-j(\omega - \omega_n)T/2}) + \\
 &\quad + \frac{|A|}{j2T(\omega + \omega_n)} (e^{j(\omega + \omega_n)T/2} - e^{-j(\omega + \omega_n)T/2}) \\
 &= \frac{|A|}{T(\omega - \omega_n)} \sin((\omega - \omega_n)T/2) + \frac{|A|}{T(\omega + \omega_n)} \sin((\omega + \omega_n)T/2).
 \end{aligned}$$

5 This can be simplified further if we substitute  $T = \pi/\omega$  and  $\omega_n = 2\pi n/T = 2n\omega$ ,



6 Using a product-to-sum trigonometric identity ([Appendix 01.2 math.trig](#)), this further simplifies to

$$c_{\pm n} = \frac{-2|A|}{\pi(4n^2 - 1)} \cos(\pi n),$$





```

• ans =
(2*abs(A))/(pi*(4*n^2 - 1))

• ans =
-(2*abs(A))/(pi*(4*n^2 - 1))

```

11 These are also what we got before.

### Parb b: harmonic amplitude and phase with spectra

12 According to Eq. 12, the harmonic amplitude is

$$\begin{aligned}
 C_n &= 2\sqrt{c_n c_{-n}} \\
 &= \frac{4|A|}{\pi|4n^2 - 1|} |\cos(\pi n)|
 \end{aligned}$$

13 Let's check with Matlab.

```

assume(n, 'real');
C_n = simplify(2*sqrt(c_n*subs(c_n,n,-n)))
assume(n, 'clear');
assume(n, 'integer');
C_n = simplify(2*sqrt(c_n*subs(c_n,n,-n)))

```

```

C_n =

(4*abs(A)*abs(cos(pi*n)))/(pi*abs(4*n^2 - 1))

C_n =

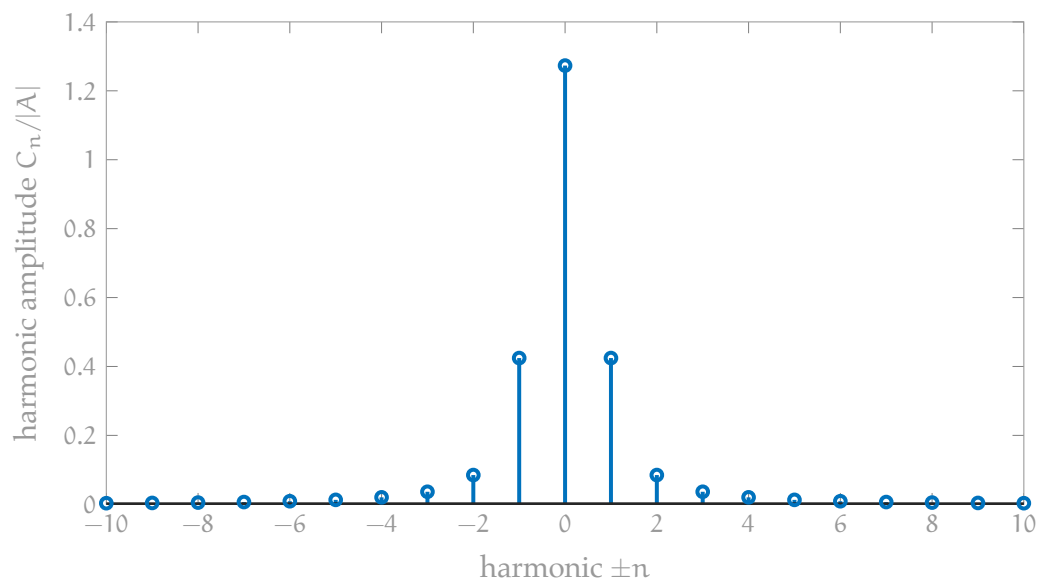
(4*abs(A))/(pi*abs(4*n^2 - 1))

```

14 We see that if we assume  $n$  is an integer,  $C_n$  simplifies even further than we took it by-hand.

- 15 Plotting the harmonic amplitude is straightforward. First make  $C_n$  something that can be numerically evaluated and choose parameters.

```
p.A = 1;
C_n_fun = matlabFunction( ...
    subs(C_n, p) ...
);
```



**Figure fsex.2:** the harmonic amplitude  $C_n$ .

- 16 Now we plot.

```
n_a = -10:10;
figure
stem(n_a,C_n_fun(n_a))
xlabel('\pm n')
ylabel('harmonic amplitude C_n/|A|')
```

- 17 Let's find the phase *à la* Eq. 13 with Matlab directly.

```
phase_n = simplify(atan2(imag(c_n),real(c_n)))
```

```
phase_n =
```

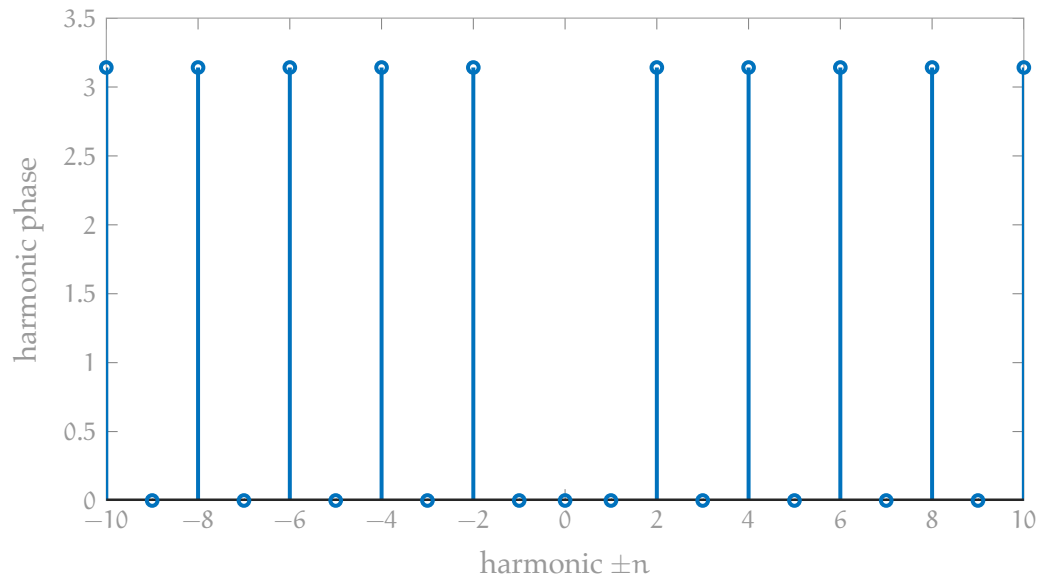
```
(pi*sign((-1)^n*abs(A))/(4*n^2 - 1))*(sign((-1)^n*abs(A))/(4*n^2 - 1) +  
↪ 1))/2
```

**18** The `sign` function just returns the sign of its argument. It's difficult to see, but this expression only takes on the following two values:



**19** We can plot the phase similarly to how we plotted the amplitude. First we get a numerically evaluable function.

```
phase_n_fun = matlabFunction( ...  
    subs(phase_n, p) ...  
);
```



**Figure fsex.a.3:** the harmonic phase.

20 Now we plot.

```
figure
stem(n_a, phase_n_fun(n_a))
xlabel('\pm n')
ylabel('harmonic phase')
```


### Part c: conversion to trig form

21 According to [Definition 09 four.3](#), the trigonometric components can be computed from the complex components as follows.

```
a_n = simplify(c_n + subs(c_n, n, -n))
b_n = simplify(j*(c_n - subs(c_n, n, -n)))
```

```
a_n =
```

```
-(4*(-1)^n*abs(A))/(pi*(4*n^2 - 1))
```


$$b_n = 0$$

**22** The fact that  $b_n = 0$  should not surprise us:  $f(t)$  is *even* after all!

## 09.3 four.transform Fourier transform

We begin with the usual loading of modules.

```
import numpy as np # for numerics
import sympy as sp # for symbolics
import matplotlib.pyplot as plt # for plots!
from IPython.display import display, Markdown, Latex
```

Let's consider a periodic function  $f$  with period  $T$  ( $T$ ). Each period, the function has a triangular pulse of width  $\delta$  (`pulse_width`) and height  $\delta/2$ .

```
period = 15 # period
pulse_width = 2 # pulse width
```

First, we plot the function  $f$  in the time domain. Let's begin by defining  $f$ .

```
def pulse_train(t,T,pulse_width):
    f = lambda x:pulse_width/2-abs(x) # pulse
    tm = np.mod(t,T)
    if tm <= pulse_width/2:
        return f(tm)
    elif tm >= T-pulse_width/2:
        return f(-(tm-T))
    else:
        return 0
```

Now, we develop a numerical array in time to plot  $f$ .

```
N = 201 # number of points to plot
tpp = np.linspace(-period/2,5*period/2,N) # time values
fpp = np.array(np.zeros(tpp.shape))
for i,t_now in enumerate(tpp):
    fpp[i] = pulse_train(t_now,period,pulse_width)
```

<sup>1</sup>Python code in this section was generated from a Jupyter notebook named `fourier_series_to_transform.ipynb` with a python3 kernel.

```

p = plt.figure(1)
plt.plot(tpp,fpp,'b-',linewidth=2) # plot
plt.xlabel('time (s)')
plt.xlim([-period/2,3*period/2])
plt.xticks(
    [0,period],
    [0,'$T='+str(period)+'$ s']
)
plt.yticks([0,pulse_width/2],['0','$\delta/2'])
plt.show() # display here

```

For  $\delta = 2$  and  $T \in [5, 15, 25]$ , the left-hand column of Fig. transform.1 shows two triangle pulses for each period  $T$ .

Consider the following argument. Just as a Fourier series is a frequency domain representation of a periodic signal, a Fourier transform is a frequency domain representation of an *aperiodic* signal (we will rigorously define it in a moment). The Fourier series components will have an analog, then, in the Fourier transform. Recall that they can be computed by integrating over a period of the signal. If we increase that period infinitely, the function is effectively aperiodic. The result (within a scaling factor) will be the Fourier transform analog of the Fourier series components.

Let us approach this understanding by actually computing the Fourier series components for increasing period  $T$  using `??`. We'll use `sympy` to compute the Fourier series cosine and sine components  $a_n$  and  $b_n$  for component  $n$  ( $n$ ) and period  $T$  ( $T$ ).

```

sp.var('x,a_0,a_n,b_n',real=True)
sp.var('delta,T',positive=True)
sp.var('n',nonnegative=True)
# a0 = 2/T*sp.integrate(
#     (delta/2-sp.Abs(x)),
#     (x,-delta/2,delta/2) # otherwise zero
# ).simplify()
an = sp.integrate(
    2/T*(delta/2-sp.Abs(x))*sp.cos(2*sp.pi*n/T*x),
    (x,-delta/2,delta/2) # otherwise zero
).simplify()

```

```
bn = 2/T*sp.integrate(
    (delta/2-sp.Abs(x))*sp.sin(2*sp.pi*n/T*x),
    (x,-delta/2,delta/2) # otherwise zero
).simplify()
display(sp.Eq(a_n,an),sp.Eq(b_n,bn))
```

$$a_n = \begin{cases} \frac{T(1-\cos(\frac{\pi\delta n}{T}))}{\pi^2 n^2} & \text{for } n \neq 0 \\ \frac{\delta^2}{2T} & \text{otherwise} \end{cases}$$

$$b_n = 0$$

Furthermore, let us compute the *harmonic amplitude* (`f_harmonic_amplitude`):

$$C_n = \sqrt{a_n^2 + b_n^2} \quad (1)$$

which we have also scaled by a factor  $T/\delta$  in order to plot it with a convenient scale.

```
sp.var('C_n',positive=True)
cn = sp.sqrt(an**2+bn**2)
display(sp.Eq(C_n,cn))
```

$$C_n = \begin{cases} \frac{T|\cos(\frac{\pi\delta n}{T})-1|}{\pi^2 n^2} & \text{for } n \neq 0 \\ \frac{\delta^2}{2T} & \text{otherwise} \end{cases}$$

Now we lambdify the symbolic expression for a numpy function.

```
cn_f = sp.lambdify((n,T,delta),cn)
```

Now we can plot.

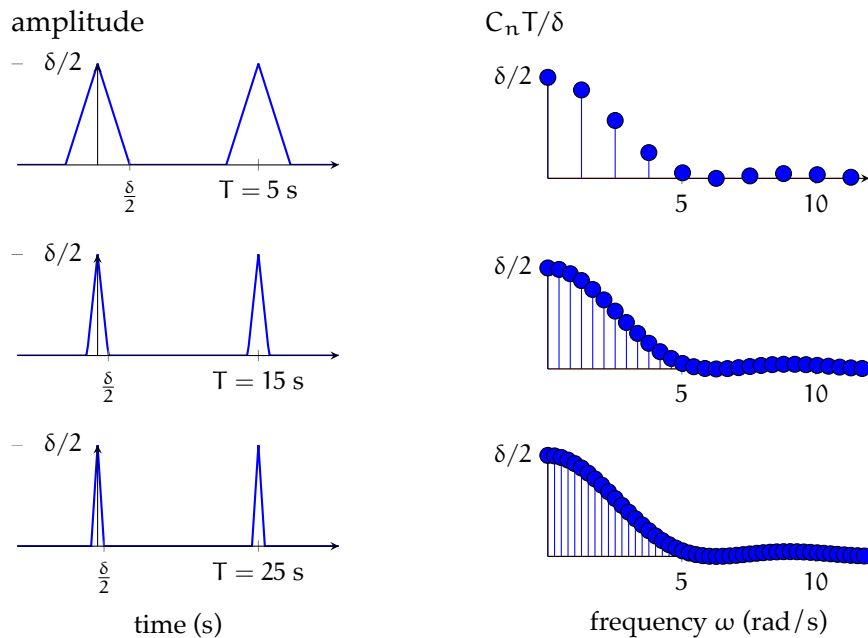
```
omega_max = 12 # rad/s max frequency in line spectrum
n_max = round(omega_max*period/(2*np.pi)) # max harmonic
n_a = np.linspace(0,n_max,n_max+1)
omega = 2*np.pi*n_a/period
p = plt.figure(2)
markerline, stemlines, baseline = plt.stem(
    omega, period/pulse_width*cn_f(n_a,period,pulse_width),
    linefmt='b-', markerfmt='bo', basefmt='r-',
```



```

    use_line_collection=True,
)
plt.xlabel('frequency  $\omega$  (rad/s)')
plt.xlim([0,omega_max])
plt.ylim([0,pulse_width/2])
plt.yticks([0,pulse_width/2],['0',' $\delta/2$ '])
plt.show() # show here

```



**Figure transform.1:** triangle pulse trains (left column) with longer periods, descending, and their corresponding line spectra (right column), scaled for convenient comparison.

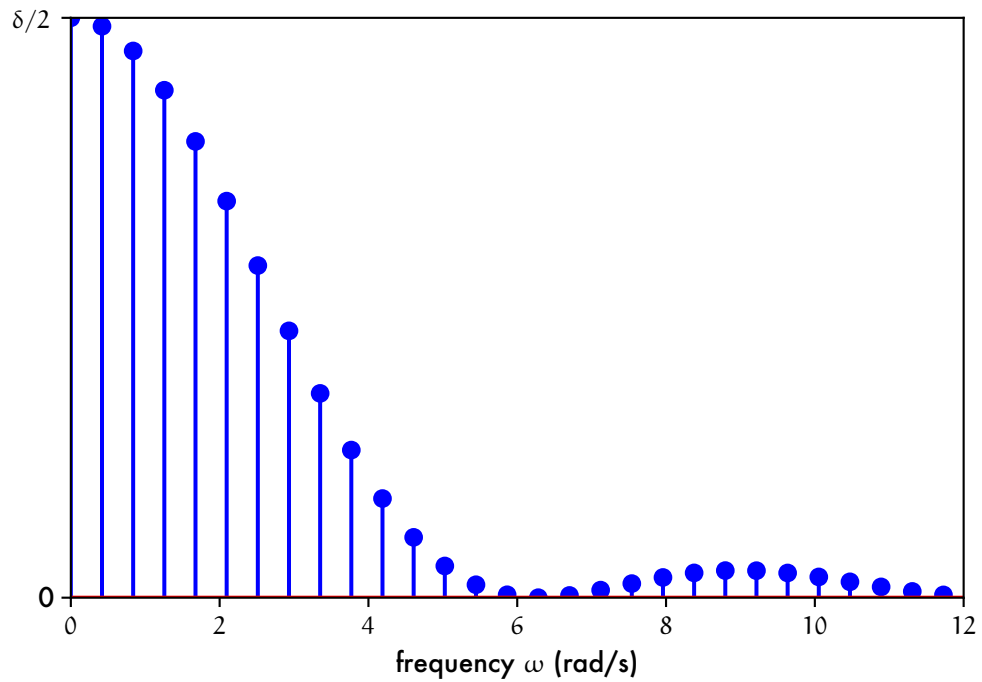
The line spectra are shown in the right-hand column of [Fig. transform.1](#). Note that with our chosen scaling, as  $T$  increases, the line spectra reveal a distinct waveform.

Let  $F$  be the continuous function of angular frequency  $\omega$

$$F(\omega) = \frac{\delta}{2} \cdot \frac{\sin^2(\omega\delta/4)}{(\omega\delta/4)^2}. \quad (2)$$

First, we plot it.

```
F = lambda w: pulse_width/2* \
    np.sin(w*pulse_width/(2*2))**2/ \
    (w*pulse_width/(2*2))**2
N = 201 # number of points to plot
wpp = np.linspace(0.0001,omega_max,N)
Fpp = []
for i in range(0,N):
    Fpp.append(F(wpp[i])) # build array of function values
axes = plt.figure(3)
plt.plot(wpp,Fpp,'b-',linewidth=2) # plot
plt.xlim([0,omega_max])
plt.yticks([0,pulse_width/2],['0','$\delta/2$'])
plt.xlabel('frequency $\omega$ (rad/s)')
plt.ylabel('$F(\omega)$')
plt.show()
```



**Figure transform.2:**  $F(\omega)$ , our mysterious Fourier series amplitude analog.

Let's consider the plot in Fig. transform.2 of  $F$ . It's obviously the function emerging in Fig. transform.1 from increasing the period of our pulse train. Now we are ready to define the Fourier transform and its inverse.

#### Definition 09 four.4: Fourier transforms: trigonometric form

Fourier transform (analysis):

$$A(\omega) = \int_{-\infty}^{\infty} y(t) \cos(\omega t) dt \quad (3)$$

$$B(\omega) = \int_{-\infty}^{\infty} y(t) \sin(\omega t) dt. \quad (4)$$

Inverse Fourier transform (synthesis):

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A(\omega) \cos(\omega t) d\omega + \frac{1}{2\pi} \int_{-\infty}^{\infty} B(\omega) \sin(\omega t) d\omega. \quad (5)$$

**Definition 09 four.5: Fourier transforms: complex form**

Fourier transform  $\mathcal{F}$  (analysis):

$$\mathcal{F}(y(t)) = Y(\omega) = \int_{-\infty}^{\infty} y(t) e^{-j\omega t} dt. \quad (6)$$

Inverse Fourier transform  $\mathcal{F}^{-1}$  (synthesis):

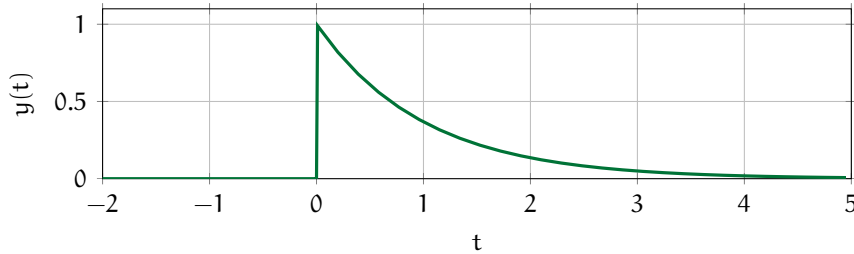
$$\mathcal{F}^{-1}(Y(\omega)) = y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega) e^{j\omega t} d\omega. \quad (7)$$

So now we have defined the Fourier transform. There are many applications, including solving differential equations and *frequency domain* representations—called *spectra*—of *time domain* functions.

There is a striking similarity between the Fourier transform and the Laplace transform, with which you are already acquainted. In fact, the Fourier transform is a special case of a Laplace transform with Laplace transform variable  $s = j\omega$  instead of having some real component. Both transforms convert differential equations to algebraic equations, which can be solved and inversely transformed to find time-domain solutions. The Laplace transform is especially important to use when an input function to a differential equation is not absolutely integrable and the Fourier transform is undefined (for example, our definition will yield a transform for neither the unit step nor the unit ramp functions). However, the Laplace transform is also preferred for *initial value problems* due to its convenient way of handling them. The two transforms are equally useful for solving steady state problems. Although the Laplace transform has many advantages, for spectral considerations, the Fourier transform is the only game in town. A table of Fourier transforms and their properties can be found in [Appendix 03.4 sum.ft.](#)

**Example 09.3 four.transform-1**re: a  
**Fourier  
transform**

Consider the aperiodic signal  $y(t) = u_s(t)e^{-at}$  with  $u_s$  the unit step function and  $a > 0$ . The signal is plotted below. Derive the complex frequency spectrum and plot its magnitude and phase.

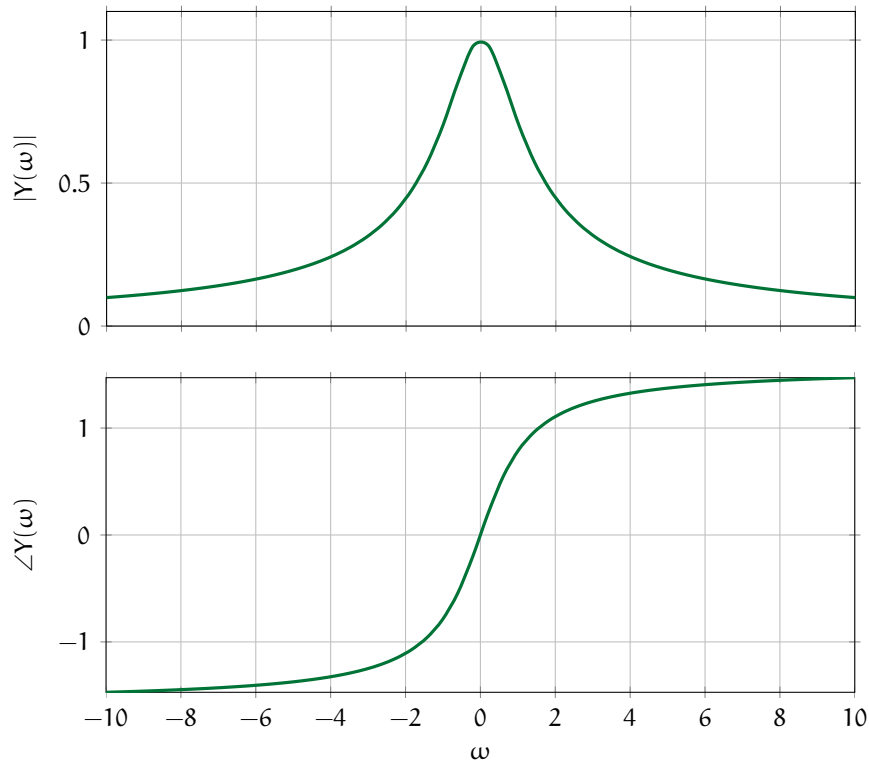


**Figure transform.3:** an aperiodic signal.

The signal is aperiodic, so the Fourier transform can be computed from Eq. 6:

$$\begin{aligned}
 Y(\omega) &= \int_{-\infty}^{\infty} y(t)e^{j\omega t} dt \\
 &= \int_{-\infty}^{\infty} u_s(t)e^{-at}e^{j\omega t} dt && \text{(def. of } y) \\
 &= \int_0^{\infty} e^{-at}e^{j\omega t} dt && \text{(} u_s \text{ effect)} \\
 &= \int_0^{\infty} e^{(-a+j\omega)t} dt && \text{(multiply)} \\
 &= \frac{1}{-a+j\omega} e^{(-a+j\omega)t} \Big|_0^{\infty} dt && \text{(antiderivative)} \\
 &= \frac{1}{-a+j\omega} \left( \lim_{t \rightarrow \infty} e^{(-a+j\omega)t} - e^0 \right) && \text{(evaluate)} \\
 &= \frac{1}{-a+j\omega} \left( \lim_{t \rightarrow \infty} e^{-at}e^{j\omega t} - 1 \right) && \text{(arrange)} \\
 &= \frac{1}{-a+j\omega} ((0)(\text{complex with mag } \leq 1) - 1) && \text{(limit)} \\
 &= \frac{-1}{-a+j\omega} && \text{(consequence)} \\
 &= \frac{1}{a-j\omega} \\
 &= \frac{a+j\omega}{a+j\omega} \cdot \frac{1}{a-j\omega} && \text{(rationalize)}
 \end{aligned}$$

$$= \frac{a + j\omega}{a^2 + \omega^2}.$$



**Figure transform.4:** the magnitude and phase of the Fourier transform.

The magnitude and phase of this complex function are straightforward to compute:

$$\begin{aligned} |Y(\omega)| &= \sqrt{\operatorname{Re}(Y(\omega))^2 + \operatorname{Im}(Y(\omega))^2} \\ &= \frac{1}{a^2 + \omega^2} \sqrt{a^2 + \omega^2} \\ &= \frac{1}{\sqrt{a^2 + \omega^2}} \\ \angle Y(\omega) &= \arctan(\omega/a). \end{aligned}$$

Now we can plot these functions of  $\omega$ . Setting  $a = 1$  (arbitrarily), we obtain the plots of Fig. transform.4.

## 09.4 four.dft Discrete and fast Fourier transforms

Modern measurement systems primarily construct spectra by sampling an analog electronic signal  $y(t)$  to yield the sample sequence  $(y_n)$  and perform a *discrete Fourier transform*.

### Definition 09 four.6: discrete Fourier transform

The *discrete Fourier transform* (DFT) of a sample sequence  $(y_n)$  of length  $N$  is  $(Y_m)$ , where  $m \in [0, 1, \dots, N - 1]$  and

$$Y_m = \sum_{n=0}^{N-1} y_n e^{-j2\pi mn/N}.$$

The *inverse discrete Fourier transform* (IDFT) reconstructs the original sequence for  $n \in [0, 1, \dots, N - 1]$  and

$$y_n = \frac{1}{N} \sum_{m=0}^{N-1} Y_m e^{j2\pi mn/N}.$$

The DFT  $(Y_m)$  has a frequency interval equal to the sampling frequency  $\omega_s/N$  and the IDFT  $(y_n)$  has time interval equal to the sampling time  $T$ . The first  $N/2 + 1$  DFT  $(Y_m)$  values correspond to frequencies

$$(0, \omega_s/N, 2\omega_s/N, \dots, \omega_s/2)$$

and the remaining  $N/2 - 1$  correspond to frequencies

$$(-\omega_s/2, -(N-1)\omega_s/N, \dots, -\omega_s/N).$$

In practice, the definitions of the DFT and IDFT are not the most efficient methods of computation. A clever algorithm called the *fast Fourier transform* (FFT) computes the DFT much more efficiently. Although it is a good exercise to roll our own FFT, in this lecture we will use `scipy`'s built-in FFT algorithm, loaded with the following command.

<sup>1</sup>Python code in this section was generated from a Jupyter notebook named `discrete_fourier_transform.ipynb` with a python3 kernel.

```
from scipy import fft
```

Now, given a time series array  $y$  representing  $(y_i)$ , the DFT (using the FFT algorithm) can be computed with the following command.

```
fft(y)
```

In the following example, we will apply this method of computing the DFT.

### Example 09.4 four.dft-1

re: FFT  
of a  
sawtooth  
signal

We would like to compute the DFT of a sample sequence  $(y_n)$  generated by sampling a spaced-out sawtooth. Let's first generate the sample sequence and plot it.

In addition to `scipy`, let's import `matplotlib` for figures and `numpy` for numerical computation.

```
import matplotlib.pyplot as plt  
import numpy as np
```

We define several "control" quantities for the spaced-sawtooth signal.

```
f_signal = 48 # frequency of the signal  
spaces = 1 # spaces between sawteeth  
n_periods = 10 # number of signal periods  
n_samples_sawtooth = 10 # samples/sawtooth
```

These quantities imply several "derived" quantities that follow.

```
••• n_samples_period = n_samples_sawtooth*(1+spaces)  
••• n_samples = n_periods*n_samples_period
```



```

T_signal = 1.0/f_signal # period of signal
t_a = np.linspace(0,n_periods*T_signal,n_samples)
dt = n_periods*T_signal/(n_samples-1) # sample time
f_sample = 1./dt # sample frequency

```

We want an interval of ramp followed by an interval of “space” (zeros). The following method of generating the sampled signal  $y$  helps us avoid *leakage*, which we’ll describe at the end of the example.

```

arr_zeros = np.zeros(n_samples_sawtooth) # frac of period
arr_ramp = np.arange(n_samples_sawtooth) # frac of period
y = [] # initialize time sequence
for i in range(n_periods):
    y = np.append(y,arr_ramp) # ramp
    for j in range(spaces):
        y = np.append(y,arr_zeros) # space

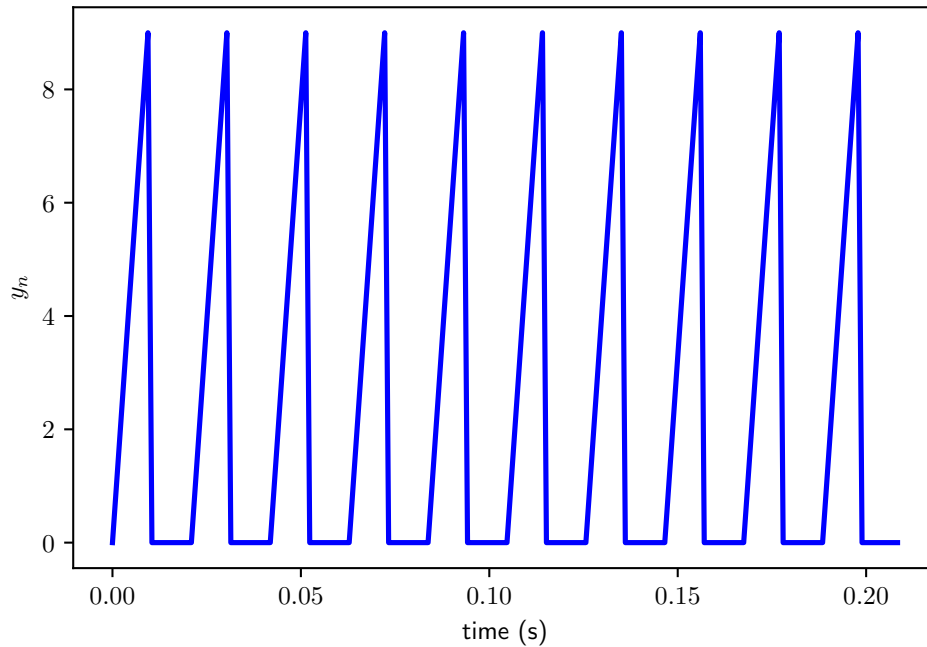
```

We plot the result in [Fig. dft.1](#), generated by the following code.

```

fig, ax = plt.subplots()
plt.plot(t_a,y,'b-',linewidth=2)
plt.xlabel('time (s)')
plt.ylabel('$y_n$')
plt.show()

```



**Figure dft.1:** the sawtooth signal in the time-domain.

Now we have a nice time sequence on which we can perform our DFT. It's easy enough to compute the FFT.

```
Y = fft(y)/n_samples # FFT with proper normalization
```

Recall that the latter values correspond to negative frequencies. In order to plot it, we want to rearrange our Y array such that the elements corresponding to negative frequencies are first. It's a bit annoying, but *c'est la vie*.

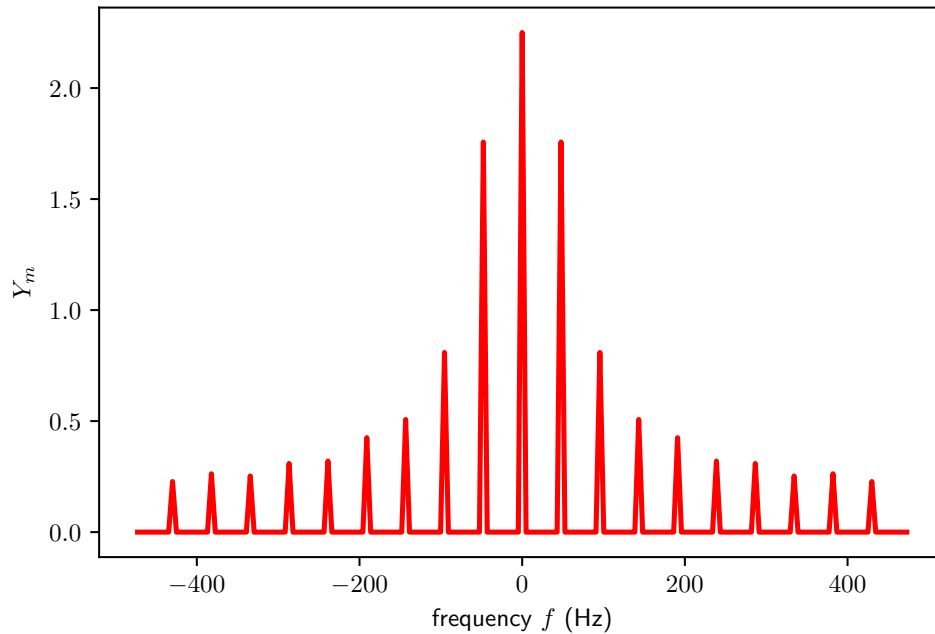
```
Y_positive_zero = Y[range(int(n_samples/2))]
Y_negative = np.flip(
    np.delete(Y_positive_zero,0),0
)
Y_total = np.append(Y_negative,Y_positive_zero)
```

• Now all we need is a corresponding frequency array.

```
freq_total = np.arange(  
    -n_samples/2+1,n_samples/2  
) * f_sample/n_samples
```

The plot, created with the following code, is shown in [Fig. dft.2](#).

```
fig, ax = plt.subplots()  
plt.plot(freq_total, abs(Y_total), 'r-', linewidth=2)  
plt.xlabel('frequency $f$ (Hz)')  
plt.ylabel('$Y_m$')  
plt.show()
```



• **Figure dft.2:** the DFT spectrum of the sawtooth function.

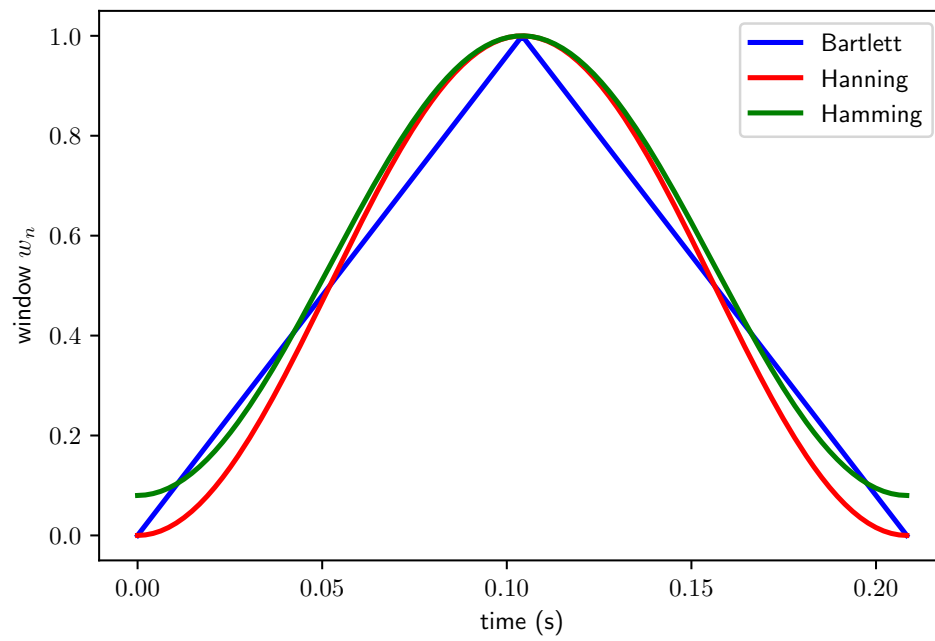
## Leakage

The DFT assumes the sequence  $(y_n)$  is periodic with period  $N$ . An implication of this is that if any periodic components have period  $N_{\text{short}} < N$ , unless  $N$  is divisible by  $N_{\text{short}}$ , spurious components will appear in  $(Y_n)$ . Avoiding leakage is difficult, in practice. Instead, typically we use a *window function* to mitigate its effects. Effectively, windowing functions—such as the *Bartlett*, *Hanning*, and *Hamming windows*—multiply  $(y_n)$  by a function that tapers to zero near the edges of the sample sequence. *Numpy* has several window functions such as `bartlett()`, `hanning()`, and `hamming()`.

Let's plot the windows to get a feel for them – see [Fig. dft.3](#).

```
bartlett_window = np.bartlett(n_samples)
hanning_window = np.hanning(n_samples)
hamming_window = np.hamming(n_samples)

fig, ax = plt.subplots()
plt.plot(t_a, bartlett_window,
         'b-', label='Bartlett', linewidth=2)
plt.plot(t_a, hanning_window,
         'r-', label='Hanning', linewidth=2)
plt.plot(t_a, hamming_window,
         'g-', label='Hamming', linewidth=2)
plt.xlabel('time (s)')
plt.ylabel('window $w_n$')
plt.legend()
plt.show()
```



**Figure dft.3:** three window functions to minimize leakage.

## 09.5 four.exe Exercises for Chapter 09 four

### Exercise 09.1 stanislaw

Explain, in your own words (supplementary drawings are ok), what the *frequency domain* is, how we derive models in it, and why it is useful.

### Exercise 09.2 pug

Consider the function

$$f(t) = 8 \cos(t) + 6 \sin(2t) + \sqrt{5} \cos(4t) + 2 \sin(4t) + \cos(6t - \pi/2).$$

(a) Find the (harmonic) magnitude and (harmonic) phase of its Fourier series components. (b) Sketch its magnitude and phase spectra. *Hint: no Fourier integrals are necessary to solve this problem.*

### Exercise 09.3 ponyo

Consider the function with  $a > 0$

$$f(t) = e^{-a|t|}.$$

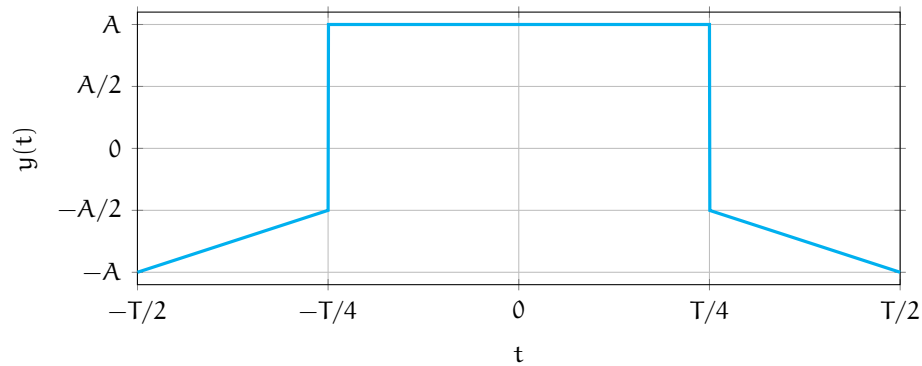
From the transform definition, derive the Fourier transform  $F(\omega)$  of  $f(t)$ . Simplify the result such that it is clear the expression is real (no imaginary component).

### Exercise 09.4 seesaw

Consider the periodic function  $f : \mathbb{R} \rightarrow \mathbb{R}$  with period  $T$  defined for one period as

$$f(t) = at \quad \text{for } t \in (-T/2, T/2] \tag{1}$$

where  $a, T \in \mathbb{R}$ . Perform a fourier series analysis on  $f$ . Letting  $a = 5$  and  $T = 1$ , plot  $f$  along with the partial sum of the fourier series synthesis, the first 50 nonzero components, over  $t \in [-T, T]$ .

**Exercise 09.5 totoro**


---

 25 p.

**Figure exe.1:** one period  $T$  of the function  $y(t)$ . Every line that appears straight is so.

Consider a periodic function  $y(t)$  with some period  $T \in \mathbb{R}$  and some parameter  $A \in \mathbb{R}$  for which one period is shown in Fig. exe.1.

1. Perform a *trigonometric* Fourier series analysis of  $y(t)$  and write the Fourier series  $Y(\omega)$ .
2. Plot the harmonic amplitude spectrum of  $Y(\omega)$  for  $A = T = 1$ . Consider using computing software.
3. Plot the phase spectrum of  $Y(\omega)$  for  $A = T = 1$ . Consider using computing software.

**Exercise 09.6 mall**

Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$f(t) = \begin{cases} a - a|t|/T & \text{for } t \in [-T, T] \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

---

 20 p.

where  $a, T \in \mathbb{R}$ . Perform a fourier transform analysis on  $f$ , resulting in  $F(\omega)$ . Plot  $F$  for various  $a$  and  $T$ .

**Exercise 09.7 miyazaki**

Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$f(t) = ae^{-b|t-T|} \quad (3)$$

where  $a, b, T \in \mathbb{R}$ . Perform a fourier transform analysis on  $f$ , resulting in  $F(\omega)$ . Plot  $F$  for various  $a, b$ , and  $T$ .

**Exercise 09.8 haku**

Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$f(t) = a \cos \omega_0 t + b \sin \omega_0 t \quad (4)$$

where  $a, b, \omega_0 \in \mathbb{R}$  constants. Perform a fourier transform analysis on  $f$ , resulting in  $F(\omega)$ .<sup>2</sup>

**Exercise 09.9 secrets**

This exercise encodes a “secret word” into a sampled waveform for decoding via a *discrete fourier transform* (DFT). The nominal goal of the exercise is to decode the secret word. Along the way, plotting and interpreting the DFT will be important.

First, load relevant packages.

```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Markdown, Latex
```

We define two functions: `letter_to_number` to convert a letter into an integer index of the alphabet (a becomes 1, b becomes 2, etc.) and `string_to_number_list` to convert a string to a list of ints, as shown in the example at the end.

<sup>2</sup>It may be alarming to see a Fourier transform of a periodic function! Strictly speaking, it does not exist; however, if we extend the transform to include the *distribution* (not actually a function) Dirac  $\delta(\omega)$ , the modified-transform does exist and is given in [Table ft.1](#).

<sup>2</sup>Python code in this section was generated from a Jupyter notebook named `random_signal_fft.ipynb` with a `python3` kernel.



```

def letter_to_number(letter):
    return ord(letter) - 96

def string_to_number_list(string):
    out = [] # list
    for i in range(0, len(string)):
        out.append(letter_to_number(string[i]))
    return out # list

print(f"aces = { string_to_number_list('aces') }")

```

```
aces = [1, 3, 5, 19]
```

Now, we encode a code string code into a signal by beginning with “white noise,” which is *broadband* (appears throughout the spectrum) and adding to it sin functions with amplitudes corresponding to the letter assignments of the code and harmonic corresponding to the position of the letter in the string. For instance, the string 'bad' would be represented by noise plus the signal

$$2 \sin 2\pi t + 1 \sin 4\pi t + 4 \sin 6\pi t. \quad (5)$$

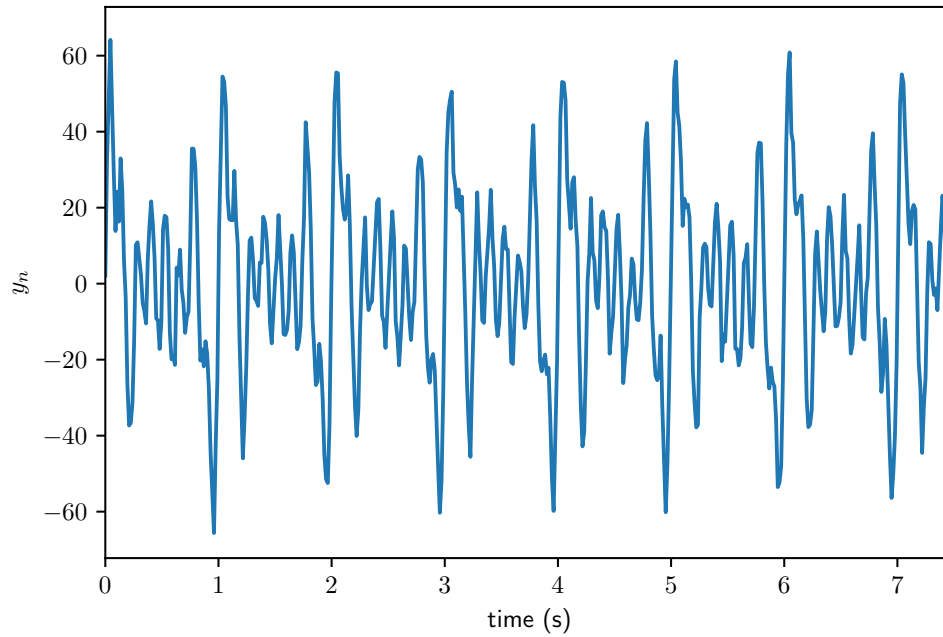
Let's set this up for secret word 'chupcabra'.

```

N = 2000
Tm = 30
T = float(Tm)/float(N)
fs = 1/T
x = np.linspace(0, Tm, N)
noise = 4*np.random.normal(0, 1, N)
code = 'chupcabra' # the secret word
code_number_array = np.array(string_to_number_list(code))
y = np.array(noise)
for i in range(0, len(code)):
    y = y + code_number_array[i]*np.sin(2.*np.pi*(i+1.)*x)

```

For proper decoding, later, it is important to know the fundamental frequency of the generated data.



**Figure exe.2:** the chupacabra signal.

```
print(f"fundamental frequency = {fs} Hz")
```

```
fundamental frequency = 66.66666666666667 Hz
```

Now, we plot.

```
fig, ax = plt.subplots()
plt.plot(x,y)
plt.xlim([0,Tm/4])
plt.xlabel('time (s)')
plt.ylabel('$y_n$')
plt.show()
```

Finally, we can save our data to a numpy file `secrets.npy` to distribute our message.

```
np.save('secrets',y)
```

Now, I have done this (for a different secret word!) and saved the data; download it here:

[ricopic.one/mathematical\\_foundations/source/secrets.npy](http://ricopic.one/mathematical_foundations/source/secrets.npy).

In order to load the .npy file into *Python*, we can use the following command.

```
secret_array = np.load('secrets.npy')
```

Your job is to (a) perform a DFT, (b) plot the spectrum, and (c) decode the message! Here are a few hints.

1. Use `from scipy import fft` to do the DFT.
2. Use a hanning window to minimize the end-effects. See `numpy.hanning` for instance. The `fft` call might then look like

```
2*fft(np.hanning(N)*secret_array)/N
```

where `N = len(secret_array)`.

3. Use only the *positive* spectrum; you can lop off the negative side and double the positive side.

### Exercise 09.10 society

Derive a fourier transform property for expressions including function

$f: \mathbb{R} \rightarrow \mathbb{R}$  for

$$f(t) \cos(\omega_0 t + \psi)$$

where  $\omega_0, \psi \in \mathbb{R}$ .

**Exercise 09.11 flapper**

Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$f(t) = au_s(t)e^{-bt} \cos(\omega_0 t + \psi) \quad (6)$$

where  $a, b, \omega_0, \psi \in \mathbb{R}$  and  $u_s(t)$  is the unit step function. Perform a fourier transform analysis on  $f$ , resulting in  $F(\omega)$ . Plot  $F$  for various  $a, b, \omega_0, \psi$  and  $T$ .

**Exercise 09.12 eastegg**

Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$f(t) = g(t) \cos(\omega_0 t) \quad (7)$$

where  $\omega_0 \in \mathbb{R}$  and  $g : \mathbb{R} \rightarrow \mathbb{R}$  will be defined in each part below. Perform a fourier transform analysis on  $f$  for each  $g$  below for  $\omega_1 \in \mathbb{R}$  a constant and consider how things change if  $\omega_1 \rightarrow \omega_0$ .

- a.  $g(t) = \cos(\omega_1 t)$
- b.  $g(t) = \sin(\omega_1 t)$

**Exercise 09.13 savage**

An instrument called a “lock-in amplifier” can measure a sinusoidal signal  $A \cos(\omega_0 t + \psi) = a \cos(\omega_0 t) + b \sin(\omega_0 t)$  at a known frequency  $\omega_0$  with exceptional accuracy even in the presence of significant noise  $N(t)$ . The workings of these devices can be described in two operations: first, the following operations on the signal with its noise,

$$f_1(t) = a \cos(\omega_0 t) + b \sin(\omega_0 t) + N(t),$$

$$f_2(t) = f_1(t) \cos(\omega_1 t) \quad \text{and} \quad f_3(t) = f_1(t) \sin(\omega_1 t). \quad (8)$$

where  $\omega_0, \omega_1, a, b \in \mathbb{R}$ . Note the relation of this operation to the Fourier transform analysis of [Exercise 09.12 four](#). The key is to know with some accuracy  $\omega_0$  such that the instrument can set  $\omega_1 \approx \omega_0$ . The second operation on the signal is an aggressive low-pass filter. The filtered  $f_2$  and  $f_3$

are called the *in-phase* and *quadrature* components of the signal and are typically given a complex representation

$$(\text{in-phase}) + j(\text{quadrature}).$$

Explain with fourier transform analyses on  $f_2$  and  $f_3$

- what  $F_2 = \mathcal{F}(f_2)$  looks like,
- what  $F_3 = \mathcal{F}(f_3)$  looks like,
- why we want  $\omega_1 \approx \omega_0$ ,
- why a low-pass filter is desirable, and
- what the time-domain signal will look like.

### Exercise 09.14 strawman

Consider again the lock-in amplifier explored in [Exercise 09.13 four..](#)

Investigate the lock-in amplifier numerically with the following steps.

- Generate a noisy sinusoidal signal at some frequency  $\omega_0$ . Include enough broadband white noise that the signal is invisible in a time-domain plot.
- Generate  $f_2$  and  $f_3$ , as described in [Exercise 09.13 four..](#)
- Apply a time-domain discrete low-pass filter to each  $f_2 \mapsto \phi_2$  and  $f_3 \mapsto \phi_3$ , such as `scipy`'s `scipy.signal.sosfiltfilt`, to complete the lock-in amplifier operation. Plot the results in time and as a complex (polar) plot.
- Perform a discrete fourier transform on each  $f_2 \mapsto F_2$  and  $f_3 \mapsto F_3$ . Plot the spectra.
- Construct a frequency domain low-pass filter  $F$  and apply it (multiply!) to each  $F_2 \mapsto F'_2$  and  $F_3 \mapsto F'_3$ . Plot the filtered spectra.
- Perform an inverse discrete fourier transform to each  $F'_2 \mapsto f'_2$  and  $F'_3 \mapsto f'_3$ . Plot the results in time and as a complex (polar) plot.
- Compare the two methods used, i.e. time-domain filtering versus frequency-domain filtering.

**10 freq**

---

## 10.1 freq.fir Frequency and impulse response

1 This lecture proceeds in three parts. First, the Fourier transform is used to derive the *frequency response function*. Second, this is used to derive the *frequency response*. Third, the frequency response for an impulse input is explored.

### Frequency response functions

2 Consider a dynamic system described by the *input-output differential equation*—with variable  $y$  representing the *output*, dependent variable time  $t$ , variable  $u$  representing the *input*, constant coefficients  $a_i, b_j$ , order  $n$ , and  $m \leq n$  for  $n \in \mathbb{N}_0$ —as:

$$\begin{aligned} \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = \\ b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u. \end{aligned} \quad (1)$$

3 The **Fourier transform**  $\mathcal{F}$  of Eq. 1 yields something interesting (assuming zero initial conditions):

$$\begin{aligned} \mathcal{F} \left( \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y \right) = \\ \mathcal{F} \left( b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u \right) \Rightarrow \\ \mathcal{F} \left( \frac{d^n y}{dt^n} \right) + a_{n-1} \mathcal{F} \left( \frac{d^{n-1} y}{dt^{n-1}} \right) + \cdots + a_1 \mathcal{F} \left( \frac{dy}{dt} \right) + a_0 \mathcal{F}(y) = \\ b_m \mathcal{F} \left( \frac{d^m u}{dt^m} \right) + b_{m-1} \mathcal{F} \left( \frac{d^{m-1} u}{dt^{m-1}} \right) + \cdots + b_1 \mathcal{F} \left( \frac{du}{dt} \right) + b_0 \mathcal{F}(u) \Rightarrow \\ (j\omega)^n Y + a_{n-1} (j\omega)^{n-1} Y + \cdots + a_1 (j\omega) Y + a_0 Y = \\ b_m (j\omega)^m U + b_{m-1} (j\omega)^{m-1} U + \cdots + b_1 (j\omega) U + b_0 U. \end{aligned}$$

Solving for  $Y$ ,



The inverse Fourier transform  $\mathcal{F}^{-1}$  of  $Y$  is the **forced response**. However, this is not our primary concern; rather, we are interested to solve for the frequency response function  $H(j\omega)$  as the ratio of the output transform  $Y$  to the input transform  $U$ , i.e.<sup>1</sup>

$$H(j\omega) \equiv \frac{Y(\omega)}{U(\omega)} \tag{2a}$$

$$= \frac{b_m(j\omega)^m + b_{m-1}(j\omega)^{m-1} + \dots + b_1(j\omega) + b_0}{(j\omega)^n + a_{n-1}(j\omega)^{n-1} + \dots + a_1(j\omega) + a_0}. \tag{2b}$$

4 Note that a frequency response function can be converted to a transfer function via the substitution  $j\omega \mapsto s$  and, conversely, a transfer function can be converted to a frequency response function<sup>2</sup> via the substitution  $s \mapsto j\omega$ , as in



It is often easiest to first derive a transfer function—using any of the methods described, previously—then convert this to a frequency response function.

### Frequency response

5 From above, we can solve for the output response  $y$  from the frequency response function by taking the inverse Fourier transform:

$$y(t) = \mathcal{F}^{-1}Y(\omega). \tag{3}$$

<sup>1</sup>It is traditional to use the non-standard, single-sided Fourier transform for the frequency response function for  $H(j\omega)$ . The motivation is that it then pairs well with the (single-sided) Laplace transform's transfer function.

<sup>2</sup>A caveat is that  $H(j\omega) = H(s)|_{s \mapsto j\omega}$  only holds if the corresponding single-sided Fourier transform exists.



From the definition of the frequency response function (2a),

$$y(t) = \mathcal{F}^{-1}(H(j\omega)U(\omega)). \tag{4}$$

6 The **convolution theorem** states that, for two functions of time  $h$  and  $u$ ,

$$\mathcal{F}(h * u) = \mathcal{F}(h)\mathcal{F}(u) \tag{5a}$$

$$= H(j\omega)U(\omega), \tag{5b}$$

where the **convolution operator**  $*$  is defined by

$$(h * u)(t) \equiv \int_{-\infty}^{\infty} h(\tau)u(t - \tau) d\tau. \tag{6}$$

Therefore,

$$y(t) = \int_{-\infty}^{\infty} H(j\omega)U(\omega)e^{j\omega t} d\omega$$

(from (5b))

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(\tau)u(t - \tau) d\tau e^{j\omega t} d\omega$$

(from (6))

This is the **frequency response** in terms of all time-domain functions.

### Impulse response

7 The frequency response result includes an interesting object:  $h(t)$ . What is the physical significance of  $h$ , other than its definition, as the inverse Fourier transform of  $H(j\omega)$ ?

8 Consider the singularity input  $u(t) = \delta(t)$ , an impulse. The frequency response is

$$y(t) = \int_{-\infty}^{\infty} h(\tau)\delta(t - \tau) d\tau. \tag{7}$$

The so-called **sifting property** of  $\delta$  yields

$$y(t) = h(t). \tag{8}$$

That is,  $h$  is the **impulse response**.

9 A very interesting aspect of this result is that

$$H(j\omega) = \mathcal{F}(h). \tag{9}$$

That is, the Fourier transform of the impulse response is the frequency response function. A way to estimate, via measurement, the frequency response function (and transfer function) of a system is to input an impulse, measure and fit the response, then Fourier transform it. Of course, putting in an actual impulse and fitting the response, perfectly are impossible; however, estimates using approximations remain useful.

10 It is worth noting that frequency response / transfer function estimation is a significant topic of study, and many techniques exist. Another method is described in [Lec. 10.2 freq.sin](#).

**Example 10.1 freq.fir-1**

Estimate the frequency response function  $H(j\omega)$  of a system from impulse response  $h(t)$  “data”. (We’ll generate this data ourselves, simulating a measured impulse response.) We will not attempt to find the functional form of  $H(j\omega)$ , just its “numerical” form, i.e. we’ll plot our estimate of the spectrum.

re:  
impulse  
response  
estimation  
of  
 $H(j\omega)$

Note that if we wanted to find a functional estimate of  $H(j\omega)$ , it would behoove us to use Matlab’s [System Identification Toolbox](#).

**Generate impulse response data**

We need a system to simulate to get this (supposedly “measured”) data. Let’s define a transfer function

$$H(s) = \frac{s + 20}{s^2 + 4s + 20}. \tag{10}$$

```
sys = tf([1,20],[1,4,20])

sys =
      s + 20
-----
```

```
s^2 + 4 s + 20
```

Continuous-time transfer function.

What are the poles?

```
poles = pole(sys)
```

```
poles =
```

```
-2.0000 + 4.0000i
```

```
-2.0000 - 4.0000i
```

This corresponds to a damped oscillator with natural frequency as follows.

```
abs(poles(1))
```

```
ans =
```

```
4.4721
```

Now let's find the impulse response.

```
fs = 1000; % Hz .. sampling frequency  
N = 2^12;  
t_a = 0:1/fs:(N-1)/fs;  
h_a = impulse(sys,t_a);
```

To make this seem a little more realistic as a "measurement," we should add some noise.

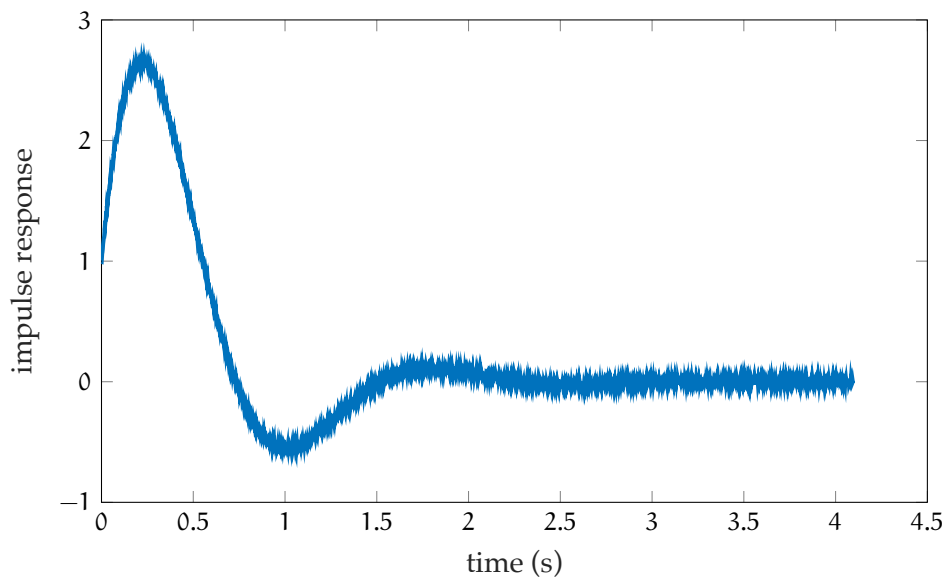
```
noise = 0.01*randn(N,1);  
h_noisy = h_a + noise;
```

• Plot the impulse response.

```

figure
plot(...
    t_a,h_noisy, ...
    'linewidth',1.5 ...
)
xlabel('time (s)')
ylabel('impulse response')

```



### Discrete Fourier transform

The discrete Fourier transform will give us an estimate of the frequency spectrum of the system; that is, a numerical version of  $H(j\omega)$ .

```
H = fft(h_noisy);
```

Compute the one-sided magnitude spectrum.

```

H_mag = abs(H/fs); % note the scaling
H_mag = H_mag(1:N/2+1); % first half, only

```

• Compute the one-sided phase spectrum.

```
H_pha = angle(H); % note the scaling
H_pha = H_pha(1:N/2+1); % first half, only
```

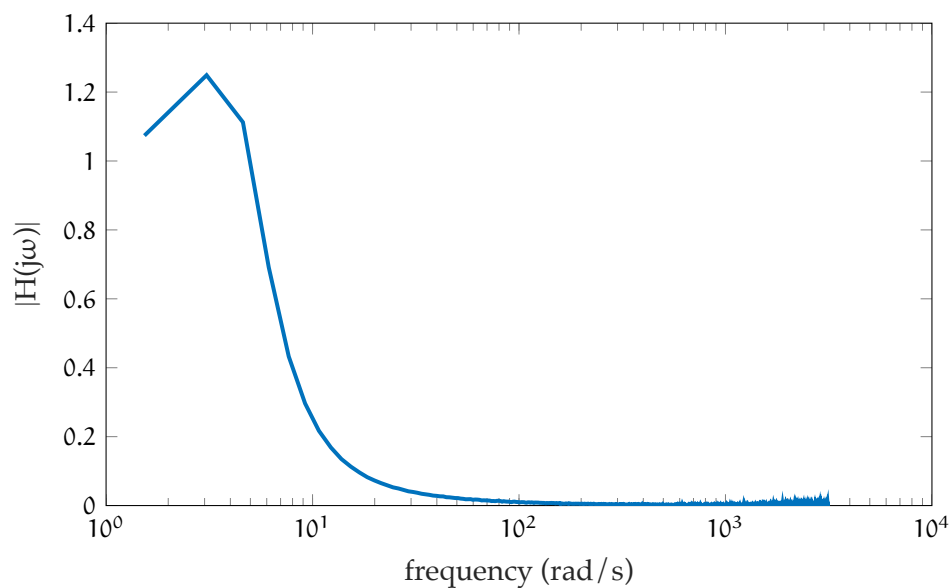
Now the corresponding frequencies.

```
f = fs*(0:(N/2))/N;
```

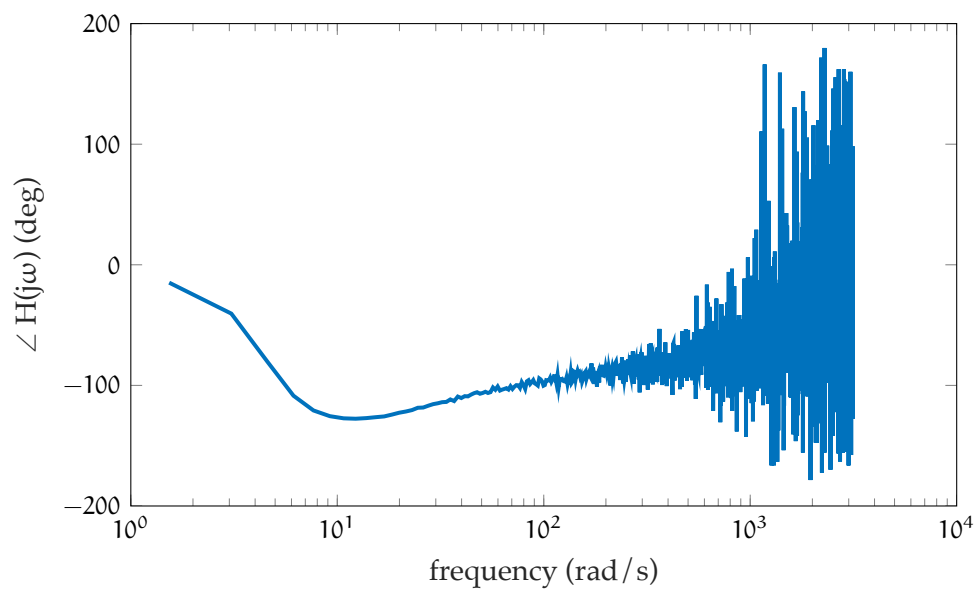
### Plot the frequency response function

We like to use a logarithmic scale, at least in frequency, for the spectrum plots.

```
figure
semilogx(...
    2*pi*f,H_mag, ...
    'linewidth',1.5 ...
)
xlabel('frequency (rad/s)')
ylabel('|H(j\omega)|')
```



```
figure
semilogx(...
    2*pi*f,180/pi*H pha, ...
    'linewidth',1.5 ...
)
xlabel('frequency (rad/s)')
ylabel('\angle H(j\omega) (deg)')
```



When the magnitude  $|H(j\omega)|$  is small, the signal-to-noise ratio is so low that the phase estimates are dismal. This can be mitigated by increasing sample-size and using more advanced techniques for estimating  $H(j\omega)$ , such as those available in Matlab's System Identification Toolbox.

## 10.2 freq.sin Sinusoidal input, frequency response

- 1 In this lecture, we explore the relationship—which turns out to be pretty chummy—between a system’s frequency response function  $H(j\omega)$  and its sinusoidal forced response.
- 2 Let’s build from the frequency response function  $H(j\omega)$  definition:

$$y(t) = \mathcal{F}^{-1}Y(\omega) \quad (1a)$$

$$= \mathcal{F}^{-1}(H(j\omega)U(\omega)). \quad (1b)$$

We take the input to be sinusoidal, with amplitude  $A \in \mathbb{R}$ , angular frequency  $\omega_0$ , and phase  $\psi$ :

$$u(t) = A \cos(\omega_0 t + \psi). \quad (2)$$

The Fourier transform of the input,  $U(\omega)$ , can be constructed via transform identities from [Table ft.1](#). This takes a little finagling. Let

$$p(t) = Aq(t), \quad (3a)$$

$$q(t) = r(t - t_0), \text{ and} \quad (3b)$$

$$r(t) = \cos \omega_0 t, \text{ where} \quad (3c)$$

$$t_0 = -\psi/\omega_0. \quad (3d)$$

The corresponding Fourier transforms, from [Table ft.1](#), are

$$P(\omega) = AQ(\omega), \quad (4a)$$

$$Q(\omega) = e^{-j\omega t_0}R(\omega), \text{ and} \quad (4b)$$

$$R(\omega) = \pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0). \quad (4c)$$

Putting these together,

(because  $\delta$ s)

3 And now we are ready to substitute into Eq. 1b; also applying the “linearity” property of the Fourier transform:

$$y(t) = A\pi (e^{j\psi}\mathcal{F}^{-1}(H(j\omega)\delta(\omega - \omega_0)) + e^{-j\psi}\mathcal{F}^{-1}(H(j\omega)\delta(\omega + \omega_0))). \quad (5)$$

The definition of the inverse Fourier transform gives

$$y(t) = \frac{A}{2} \left( e^{j\psi} \int_{-\infty}^{\infty} e^{j\omega t} H(j\omega) \delta(\omega - \omega_0) d\omega + e^{-j\psi} \int_{-\infty}^{\infty} e^{j\omega t} H(j\omega) \delta(\omega + \omega_0) d\omega \right). \quad (6)$$

Recognizing that  $\delta$  is an even distribution ( $\delta(t) = \delta(-t)$ ) and applying the sifting property of  $\delta$  allows us to evaluate each integral:

$$y(t) = \frac{A}{2} (e^{j\psi} e^{j\omega_0 t} H(j\omega_0) + e^{-j\psi} e^{-j\omega_0 t} H(-j\omega_0)). \quad (7)$$

Writing  $H$  in polar form,

(8)

The Fourier transform is **conjugate symmetric**—that is,  $F(-\omega) = F^*(\omega)$ —which allows us to further simplify:

$$y(t) = \frac{A|H(j\omega_0)|}{2} (e^{j(\omega_0 t + \psi)} e^{j\angle H(j\omega_0)} + e^{-j(\omega_0 t + \psi)} e^{-j\angle H(j\omega_0)}) \quad (9a)$$

$$= A|H(j\omega_0)| \frac{e^{j(\omega_0 t + \psi + \angle H(j\omega_0))} + e^{-j(\omega_0 t + \psi + \angle H(j\omega_0))}}{2}. \quad (9b)$$

Finally, Euler’s formula yields something that deserves a box.

Equation 10 sinusoidal response in terms of  $H(j\omega)$

4 This is a remarkable result. For an input sinusoid, a linear system has a forced response that



- is also a sinusoid,
- is at the same frequency as the input,
- differs only in amplitude and phase,
- differs in amplitude by a factor of  $|H(j\omega)|$ , and
- differs in phase by a shift of  $\angle H(j\omega)$ .

Now we see one of the key facets of the frequency response function: it governs how a sinusoid transforms through a system. And just think how powerful it will be once we combine it with the powerful principle of superposition and the mighty Fourier series representation of a function—as a “superposition” of sinusoids!

## 10.3 freq.bode Bode plots

- 1 This lecture also appears in *Control: an introduction*.
- 2 Given Eq. 10, we are often most-interested in the magnitude  $|H(j\omega)|$  and phase  $\angle H(j\omega)$  of the frequency response function. Each of these is a function of angular frequency  $\omega$ , so plotting  $|H(j\omega)|$  vs.  $\omega$  and  $\angle H(j\omega)$  vs.  $\omega$  is quite useful. **Bode plots** are such plots with axes scaled in a specific manner.
- 3 A Bode plot is a useful graphical representation of the frequency response of a system. Let  $|U(\omega)|$  and  $|Y(\omega)|$  be the complex amplitudes of the input and the output, respectively. Recall that the magnitude of the frequency response function  $|H(j\omega)|$  can be expressed as

**Equation 1 frequency response function as an amplitude ratio**

- 4 This is a ratio of amplitudes, and so it is akin to amplitude ratios commonly expressed in **decibels** (dB). However, the magnitude ratio of Eq. 1 is *not dimensionless*, and therefore cannot be expressed as decibel in the strict sense. Nevertheless, it is standard usage in system dynamics and control theory use the familiar formula to compute the logarithmic magnitude

**Equation 2 logarithmic magnitude of  $H(j\omega)$  in "dB"**

- 5 The phase is usually plotted in degrees, and the  $\omega$ -axis is logarithmic in both plots. The two plots are typically tiled vertically with the magnitude plot above the phase. We now work a simple example.

**Example 10.3 freq.bode-1**

Let a system have transfer function  $H(s) = s$ , a single zero at the origin. Find the frequency response function and draw the Bode plot for the system.

re: A  
simple  
Bode  
plot

## 10.4 freq.bodesimp Bode plots for simple transfer functions

- 1 This lecture also appears in *Control: an introduction*.
- 2 It turns out that bode plots, both magnitude and phase, given their logarithmic scale (recall that the  $\omega$ -axes are also plotted logarithmically), are quite asymptotic to straight-lines for first- and second-order systems. Furthermore, higher-order system transfer functions can be re-written as the *product* of those of first-and second-order. For instance,

$$H(s) = \frac{\underline{c}s + \underline{d}}{s^3 + \underline{a}s^2 + \underline{b}s + \underline{c}} \quad (1a)$$

$$= \underline{c} \cdot (\underline{c}s + 1) \cdot \frac{1}{\underline{c}s + 1} \cdot \frac{1}{s^2 + \underline{a}s + \underline{c}} \quad (1b)$$

- 3 Recall (from, for instance, phasor representation) that for products of complex numbers, *phases*  $\phi_i$  *add* and *magnitudes*  $M_i$  *multiply*. For instance,

$$M_1 \angle \phi_1 \cdot \frac{1}{M_2 \angle \phi_2} \cdot \frac{1}{M_3 \angle \phi_3} = \frac{M_1}{M_2 M_3} \angle (\phi_1 - \phi_2 - \phi_3). \quad (2)$$

And if one takes the logarithm of the magnitudes, they add; for instance,

$$\log \frac{M_1}{M_2 M_3} = \log M_1 - \log M_2 - \log M_3. \quad (3)$$

There is only one more link in the chain: first- and second-order Bode plots depend on a handful of parameters that can be found *directly from transfer functions*. There is no need to compute  $|H(j\omega_0)|$  and  $\angle H(j\omega_0)$ !

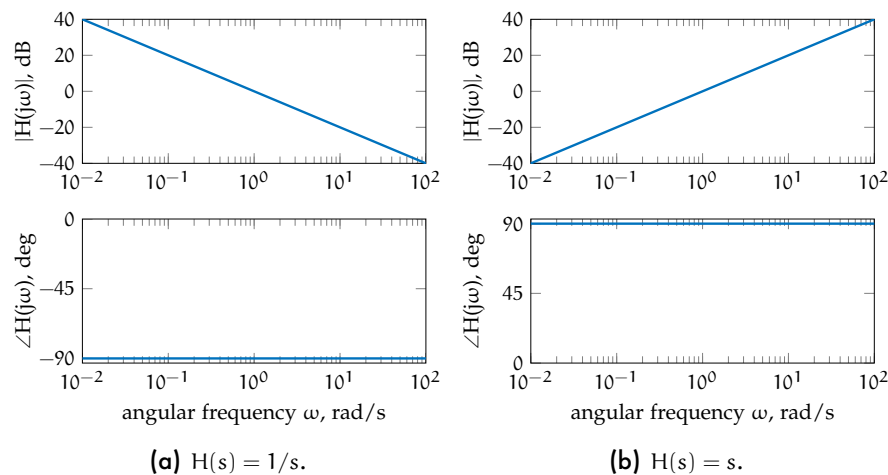
- 4 In a manner similar to [Example 10.3 freq.bode-1](#), we construct Bode plots for several simple transfer functions in this lecture. Once we have these simple “building blocks,” we will be able to construct sketches of higher-order systems by graphical addition because logarithmic magnitudes and phases combine by summation, as shown in [Lec. 10.5 freq.bodesketch](#).

*Constant gain*

5 For a transfer function that is simply a constant real gain  $H(s) = K$ , the frequency response function is trivially  $H(j\omega) = K$ . Its magnitude  $|H(j\omega)| = |K|$ . For positive gain  $K$ , the phase is  $\angle H(j\omega) = 0$ , and for negative  $K$ , the phase is  $\angle H(j\omega) = 180$  deg.

*Pole and zero at the origin*

6 In Example 10.3 [freq.bode-1](#), we have already demonstrated how to derive from the transfer function  $H(s) = s$ , a zero at the origin, the frequency response function plotted in [Fig. bodesimp.1](#). Similarly, for  $H(s) = 1/s$ , a pole at the origin, the frequency response function plotted in [Fig. bodesimp.1](#).

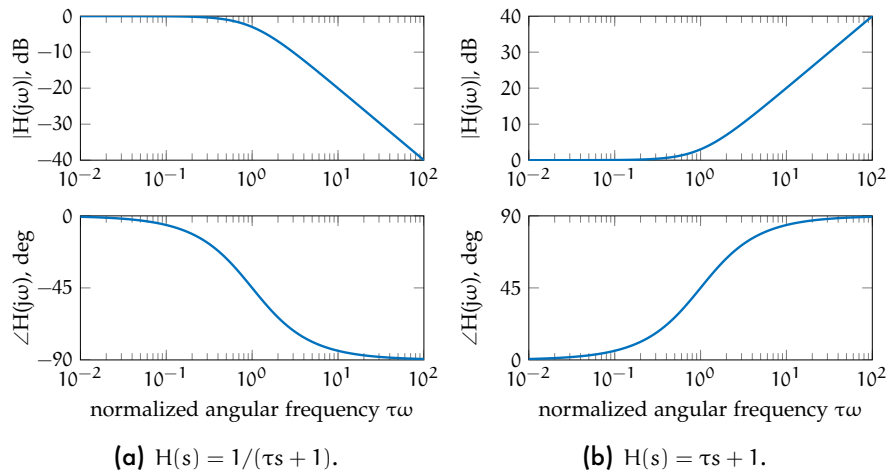


**Figure bodesimp.1:** Bode plots for (a) a pole at the origin and (b) a zero at the origin.

*Real pole and real zero*

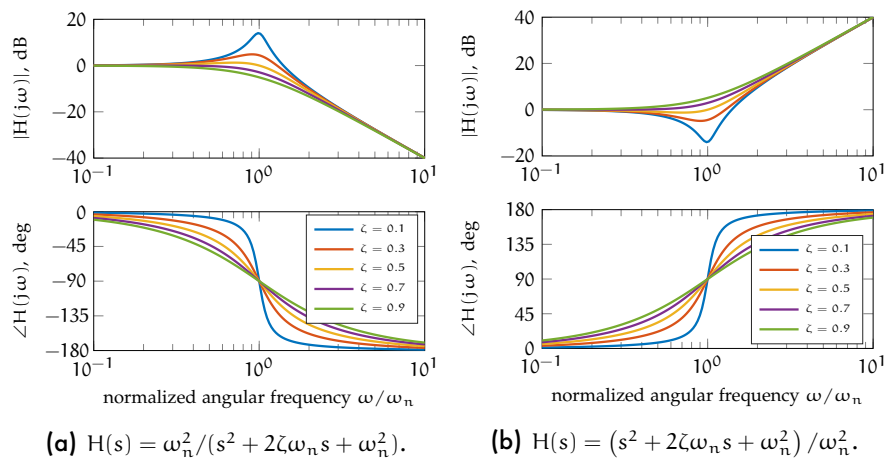
7 The derivations for real poles and zeros are not included, but the resulting Bode plots are shown in [Fig. bodesimp.2](#).

*Complex conjugate pole pairs and zero pairs*



**Figure bodesimp.2:** Bode plots for (a) a single real pole and (b) a single real zero.

8 The derivations for complex conjugate pole pairs and zero pairs are not included, but the resulting Bode plots are shown in Fig. bodesimp.3.



**Figure bodesimp.3:** Bode plots for (a) a complex conjugate pole pair and (b) a complex conjugate zero pair.

9 This lecture also appears in *Control: an introduction*.

## 10.5 freq.bodesketch Sketching Bode plots

**1** We can use MATLAB's `bode` command to create Bode plots from LTI system models. However, we must understand how these plots relate to their transfer functions. In this section, we learn to sketch Bode plots in order to deepen our intuition of this relationship.

**2** Let  $H(s) = \prod_i H_i(s)$ ; that is, let  $H(s)$  be the product of several factors  $H_i(s)$ . The magnitude and phase are

$$|H(s)| = \prod_i |H_i(s)| \quad \text{and} \quad \angle H(s) = \sum_i \angle H_i(s). \quad (1)$$

The Bode plot consists of plots of  $20 \log_{10}|H(s)|$  and  $\angle H(s)$  with  $s \mapsto j\omega$ . The magnitude and phase expressions, become

$$20 \log_{10}|H(j\omega)| = \sum_i 20 \log_{10}|H_i(j\omega)| \quad \text{and} \quad \angle H(j\omega) = \sum_i \angle H_i(j\omega). \quad (2)$$

This result means we can graphically sum both the magnitude and phase Bode plots of the individual factors of  $H(s)$ , as long as we are adding magnitudes in dB.

### Example 10.5 freq.bodesketch-1

Given the transfer function

$$H(s) = \frac{200000(s+1)}{s^3 + 110s^2 + 11000s + 100000}$$

answer the following questions and imperatives.

- Sketch a Bode plot on [Fig. bodesketch.1](#).
- Confirm the accuracy of the sketch in Matlab, using the functions `bode` and `tf`.
- If the input to a system with this transfer function is  $5 \sin(\omega t + \pi/7)$ , what is the output amplitude and phase for
  - $\omega = 1 \text{ rad/s}$ ,

re: a  
transfer  
function  
under  
analysis

ii  $\omega = 10$  rad/s, and

iii  $\omega = 1000$  rad/s?

Use Matlab's function `evalfr` to perform the calculations.

*a*

To sketch the transfer function, we must decompose the transfer function into multiple simple factors. First, we can find the poles:

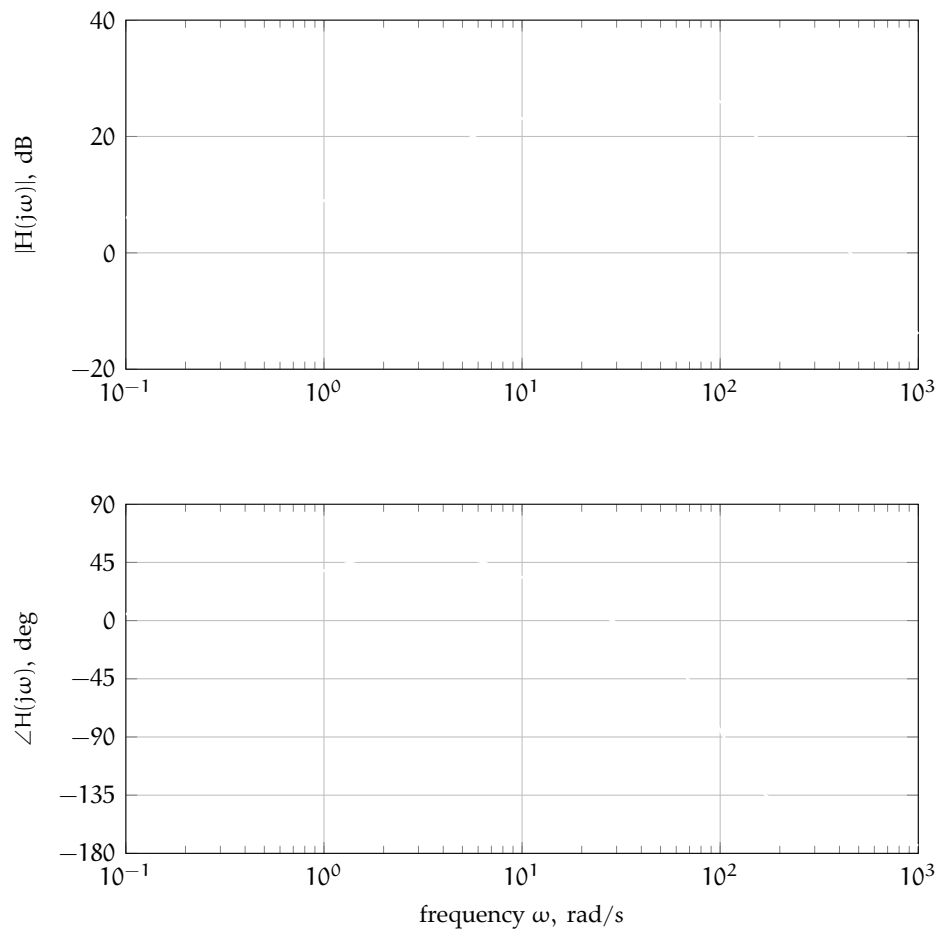
$$-10, -50 + j86.6, -50 - j86.6,$$

which tells us we have a complex conjugate pair and a single real pole. Factoring, accordingly,

$$\begin{aligned} H(s) &= 200000(s+1) \cdot \frac{1}{s+1} \cdot \frac{1}{s^2 + 100s + 10000} \\ &= 2(s+1) \cdot \frac{1}{s/10 + 1} \cdot \frac{100^2}{s^2 + 2 \cdot 0.5 \cdot 100s + 100^2} \end{aligned}$$

The sketch is shown in Fig. [bodesketch.1](#).





**Figure bodesketch.1:** a Bode plot for [Example 10.5 freq.bodesketch-1](#).

*b*

See the code listing below.

```
sys = 2e5*...  
tf(...  
    [1,1],...  
    [1,110,11000,1e5]...  
);  
bode(sys);
```

c

The output amplitude is always  $5|H(j\omega)|$  and output phase is always  $\pi/7 + \angle H(j\omega)$ . We could estimate them from the Bode plot sketch, but we instead choose to evaluate the Matlab transfer function, as in the listing below.

```

in_amp = 5;
in_phase = pi/7; % rad
omega_a = [1,10,1e3]; % rad/s
for i = 1:length(omega_a)
    H_eval = evalfr(sys,j*omega_a(i));
    H_mag = abs(H_eval);
    H_phase = angle(H_eval);
    out_amp = 5*H_mag;
    out_phase = in_phase + H_phase;
    sprintf(...
        ['For input angular freq %0.2g,\n',...
         ' input amplitude %0.2g,\n',...
         ' input phase %0.2g,\n',...
         ' H magnitude %0.2g, and\n',...
         ' H phase %0.2g,\n',...
         ' the output amplitude is %0.2g and\n',...
         ' the output phase is %0.2g.\n'...
        ],...
        omega_a(i),...
        in_amp,...
        in_phase,...
        H_mag,...
        H_phase,...
        out_amp,...
        out_phase...
    )
end

```

The output amplitudes are 14, 71, and 1 and the output phases are 1.1, 1,  $-2.6$  rad.

## 10.6 freq.per Periodic input, frequency response

1 Let a system  $H$  have a periodic input  $u$  represented by a Fourier series. For reals  $a_0$ ,  $\omega_1$  (fundamental frequency),  $A_n$ , and  $\phi_n$ , let

$$u(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} A_n \sin(n\omega_1 t + \phi_n). \quad (1)$$

The  $n$ th harmonic is

which, from Equation 10 yields forced response

2 Applying the principle of superposition, the forced response of the system to periodic input  $u$  is

$$y(t) = \frac{a_0}{2} H(j0) + \sum_{n=1}^{\infty} A_n |H(jn\omega_1)| \sin(n\omega_1 t + \phi_n + \angle H(jn\omega_1)). \quad (2)$$

3 Similarly, for inputs expressed as a complex Fourier series with components

$$u_n(t) = c_n e^{jn\omega_1 t}, \quad (3)$$

each of which has output

$$y_n(t) = c_n H(jn\omega_1) e^{jn\omega_1 t}, \quad (4)$$

the principle of superposition yields

$$y(t) = \sum_{n=-\infty}^{\infty} c_n H(jn\omega_1) e^{jn\omega_1 t}. \quad (5)$$

4 Eqs. 2 and 5 tell us that, for a periodic input, we obtain a periodic output with each harmonic  $\omega_n$  amplitude scaled by  $|H(j\omega_n)|$  and phase offset by  $\angle H(j\omega_n)$ . As a result, the response will usually undergo significant *distortion*, called **phase distortion**. The system  $H$  can be considered to **filter** the input by amplifying and suppressing different harmonics. This is why systems not intended to be used as such are still sometimes called “filters.” This way of thinking about systems is very useful to the study of vibrations, acoustics, measurement, and electronics.

5 All this can be visualized via a Bode plot, which is a significant aspect of its analytic power. An example of such a visualization is illustrated in Figure per.1.

### Example 10.6 freq.per-1

In Example 09.1 four.series-1, we found that a square wave of amplitude one has trigonometric Fourier series components

$$a_n = 0 \quad \text{and} \quad b_n = \frac{2}{n\pi} (1 - \cos(n\pi)) = \begin{cases} 0 & n \text{ even} \\ \frac{4}{n\pi} & n \text{ odd.} \end{cases}$$

Therefore, from the definitions of  $C_n$  and  $\phi_n$ , with  $b_n \geq 0$ ,

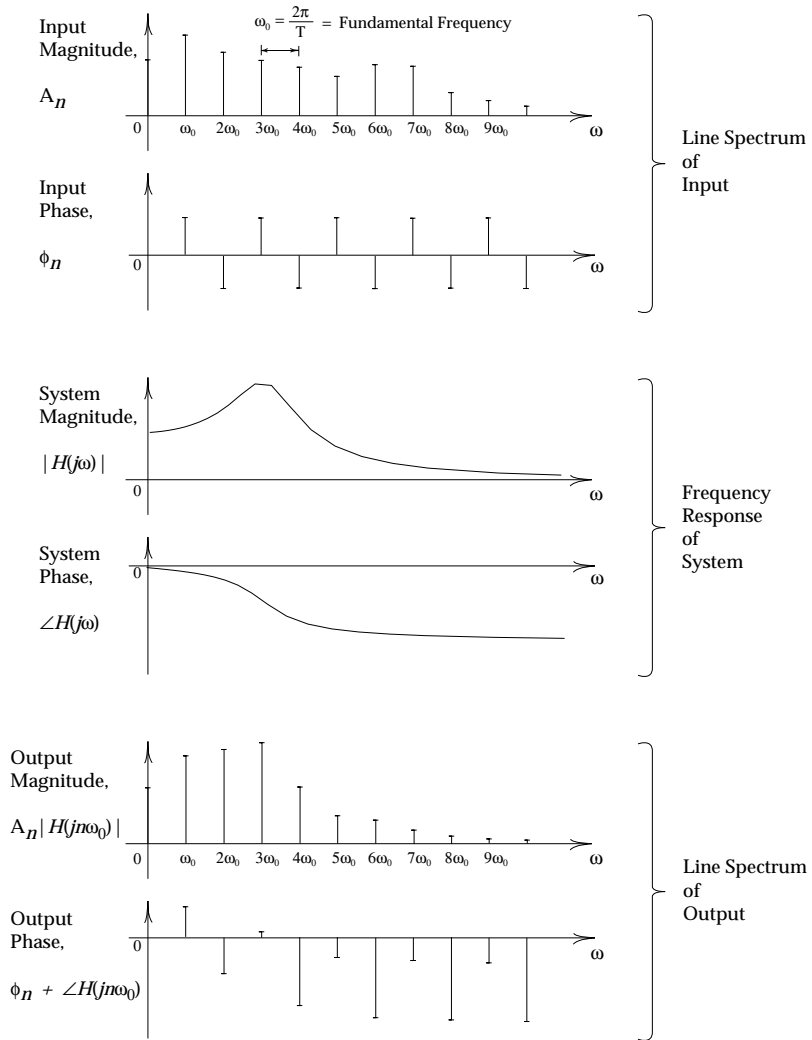
$$C_n = b_n \quad \text{and} \quad \phi_n = \arctan \frac{b_n}{a_n} = \begin{cases} i & (\text{indeterminate}) \text{ for } n \text{ even} \\ \pi/2 & \text{for } n \text{ odd.} \end{cases}$$

Let this square wave be the input  $u$  to a second-order system with frequency response function  $H(j\omega)$ , natural frequency  $\omega_N = \omega_5$  (fifth harmonic frequency), and damping ratio  $\zeta = 0.1$ .

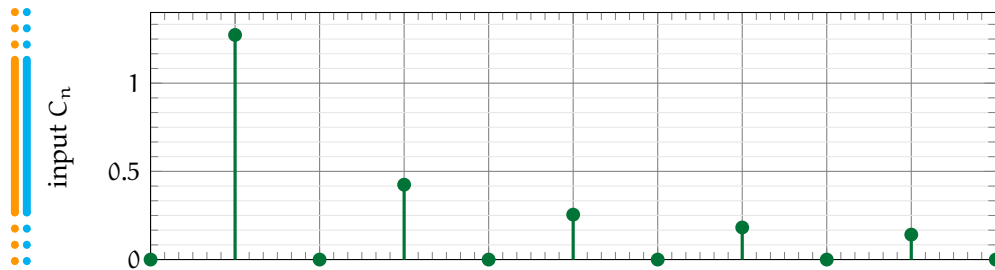
Figure per.2 and Figure per.3 show the magnitude and phase spectra for input  $u$ , frequency response function  $H(j\omega)$ , and output  $y$ .

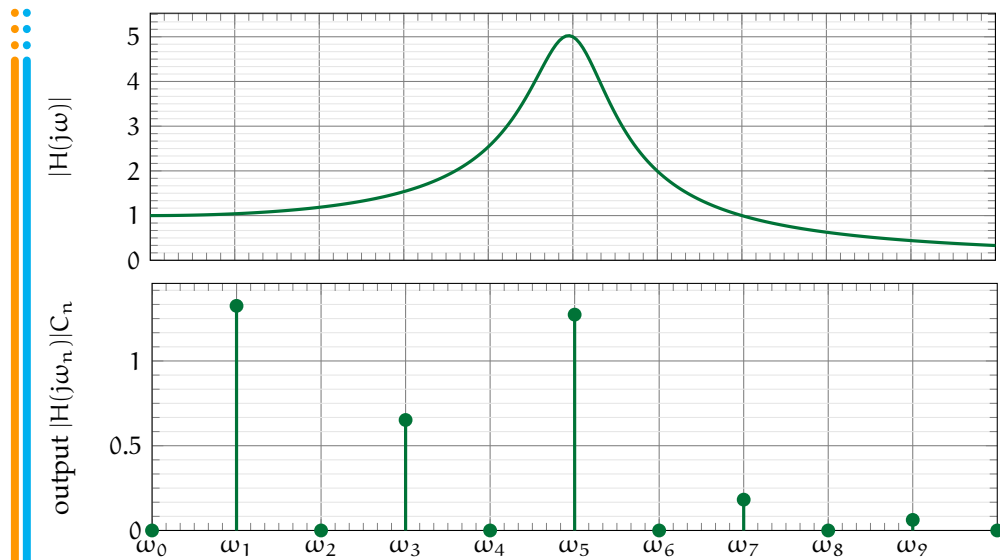
re:  
filtering  
a  
square  
wave



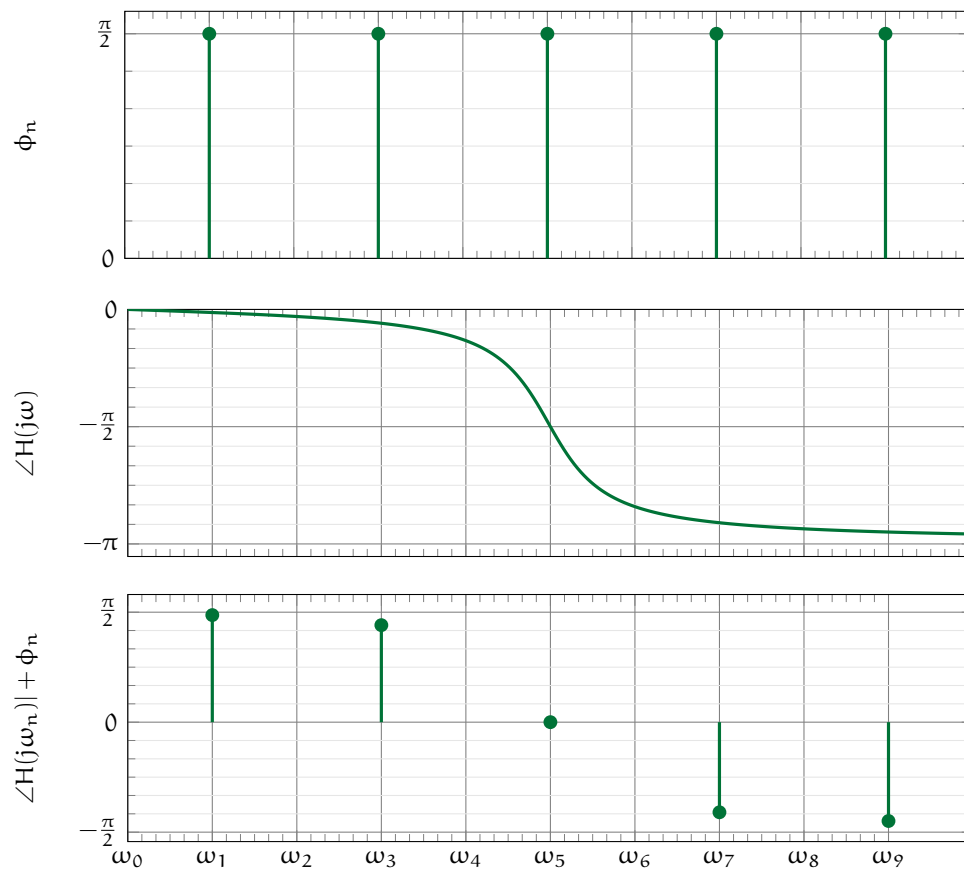


**Figure per.1:** response  $y$  of a system  $H$  to periodic input  $u$ .





**Figure per.2:** the magnitude line spectrum  $C_n$  of the input, which is operated on by the measurement system with frequency response function  $H(j\omega)$  to form the output magnitude line spectrum  $|H(j\omega_n)|C_n$ .



**Figure per.3:** the phase line spectrum  $\phi_n$  of the input, which is operated on by the measurement system with frequency response function  $H(j\omega)$  to form the output phase line spectrum  $\angle H(j\omega_n) + \phi_n$ .

## 10.7 freq.exe Exercises for Chapter 10 freq

### Exercise 10.1 gauche

Consider a system with i/o ODE

$$\ddot{y} + a \dot{y} + b y = b u \quad (1)$$

\_\_\_\_\_  
25 p.

for constants  $a, b \in \mathbb{R}$ .

1. Derive the frequency response function  $H(j\omega)$  and the transfer function  $H(s)$ . *Hint: either can be found from the other.*
2. Let  $u(t) = 7 \cos(5t + 3)$ . What is the steady state forced response  $y(t)$  in terms of  $a, b$ ? *Hint: this shouldn't require much computation.*
3. Now let  $u(t) = 3 \delta(t)$ , an impulse. What is the impulse response  $y(t)$  in terms of the inverse Fourier transform  $\mathcal{F}^{-1}$  and  $H(j\omega)$ ? Do *not* substitute in for  $H(j\omega)$  or inverse transform.
4. Use computer software to plot the Bode plot of  $H(j\omega)$  for  $a = b = 1$ .
5. For  $b = 1$ , for what range of  $a$  will there be a complex conjugate pair of poles?<sup>3</sup> *Hint: consider comparing the transfer function derived in part (a) to the standard form of the second-order transfer function in Fig. bodesimp.3a.*

### Exercise 10.2 tickle

Let a transfer function  $H$  be

$$\frac{10(s + 100)}{s^2 + 2s + 100} \quad (2)$$

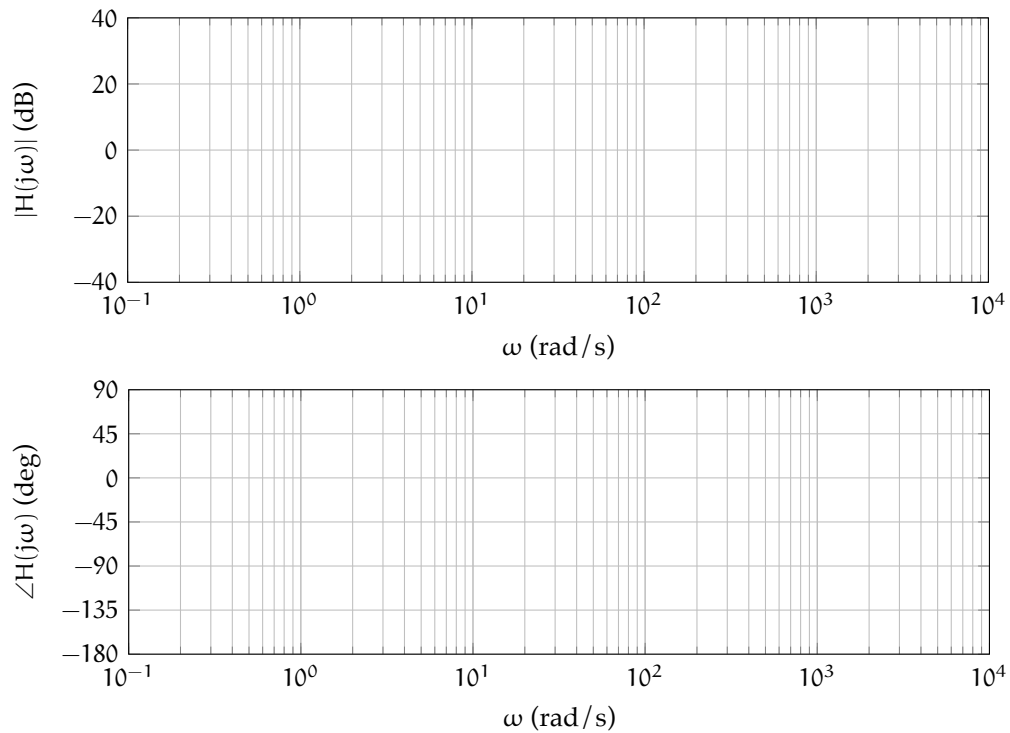
Use  $H$  to respond to the following questions and imperatives.

<sup>3</sup>The following statements are equivalent. A second-order system

- has a complex conjugate pair of poles,
- has a complex conjugate pair of the characteristic equation,
- has a complex conjugate pair of eigenvalues, and
- is underdamped.



- Write  $H$  as a product of standard-form transfer functions.
- Find the frequency response function  $H(j\omega)$  *without* simplifying.
- Use the axes below to sketch the Bode plot of  $H$ .



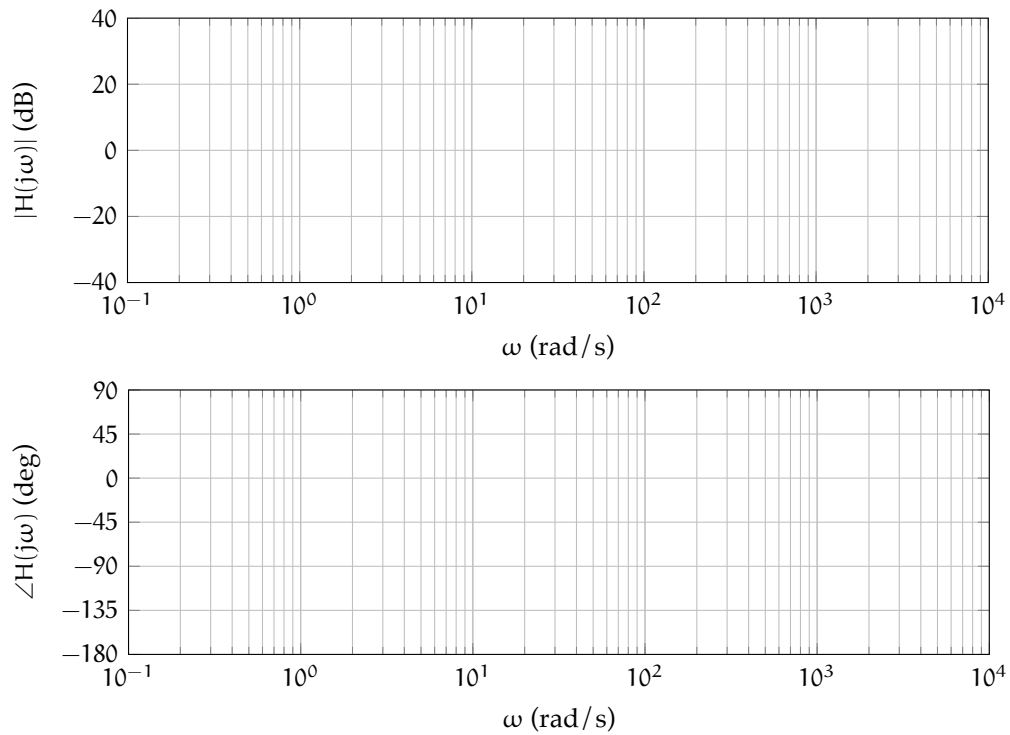
### Exercise 10.3 me

Let a transfer function  $H$  be

$$H(s) = \frac{1000(s + 10)}{(s + 100)(s + 1000)}.$$

Use  $H$  to respond to the following questions and imperatives.

- Write  $H$  as a product of standard-form transfer functions.
- Find the frequency response function  $H(j\omega)$  *without* simplifying.
- Use the axes below to sketch the Bode plot of  $H$ .



### Exercise 10.4 elmo

Consider a system with transfer function

$$H(s) = \frac{100(s + 9)}{(s + 5)(s + 6)(s^2 + 8s + 32)}.$$

- Identify the poles and zeros of  $H$ .
- Derive the frequency response function  $H(j\omega)$ . Do *not* simplify the expression.
- Create a Bode plot of  $H$ .
- Let the system have sinusoidal input  $u(t) = 2 \cos(3t)$ . What is the steady-state system output  $y(t)$ ?
- Let the system have the same sinusoidal input as previously. Simulate its forced response for nine seconds and plot it.

**Exercise 10.5 hum**

In many measurement systems, an **interference signal** (i.e., an uncontrolled, undesirable signal that appears in a signal) that often appears in measurements is the **mains hum**, which arises from the mains power grid. Its fundamental frequency is  $f_1 = 60$  Hz, and much smaller-amplitude components appear at higher harmonics.

\_\_\_\_\_/  
40 p.

Consider the following measurement system. A pressure sensor has transfer function

$$G(s) = \frac{1 \cdot 10^3}{s^2 + 1 \cdot 10^3 s + 1 \cdot 10^6}$$

with units V/Pa. The desired measurement signal has maximum angular frequency  $\omega_{\text{sig}} = 100$  rad/s. Mains hum is observed in the measurement signal with magnitude  $m = 0.5$  V.

Design a first-order low-pass filter

$$H(s) = \frac{1}{s/\omega_b + 1} \quad (3)$$

for the output of the sensor with the following steps:

- Derive the filter frequency response function  $H(j\omega)$ .
- Compute the magnitude  $|H(j\omega)|$  of the frequency response function.
- Solve for the break frequency  $\omega_b$  such that  $|H(j\omega_{\text{sig}})| = 0.97$ . This design will leave the desired signal only lightly attenuated but will further attenuate the mains hum.
- Compute the attenuation of the mains hum amplitude  $|H(j60 \cdot 2\pi)|$  and the corresponding filter output  $|H(j60 \cdot 2\pi)|m$ .
- Compute the steady-state filtered output amplitude for a sensor input  $1 \sin(\omega_{\text{sig}}t)$  kPa.
- Suppose a greater attenuation of the mains hum interference is desired. How could the filter  $H(s)$  be altered to reduce it further *without* significantly attenuating the desired signal?
- Find the sensor's natural frequency  $\omega_n$  and damping ratio  $\zeta$ . Its peak output magnitude occurs at a frequency of  $\omega_p = \omega_n \sqrt{1 - 2\zeta^2}$ . Compute  $\omega_p$  for the sensor  $G$ .

- h. Compute the steady-state filtered output amplitude for a sensor input  $1 \sin(\omega_p t)$  kPa. Explain why it is greater than the filtered output magnitude at  $\omega_{\text{sig}}$  from part e.

## **Part V**

# **Laplace analysis**

**11 lap**

---

## 11.1 lap.in Introduction

1 The Laplace transform<sup>1</sup> is a generalized Fourier transform that exists for a much broader class of functions. In fact, every function for which there is a Fourier transform, there is also a Laplace transform—but the reverse does not hold. Its excellence for linear system analysis cannot be overstated, and leads some to undervalue the Fourier transform. However, the Fourier transform is much more conceptually grounded in the frequency domain given that it can be understood as an extension of the Fourier series.

2 The Laplace transform’s conceptual grounding has the same root, but in a less-recognizable form since the explicit frequency variable  $\omega$  will be consumed by the Laplace transform  $s$ , introduced in a moment. But first, we motivate the Laplace transform by identifying a function of great importance to system analysis that does not have a Fourier transform: the unit step function  $u_s(t)$ .

3 The Fourier transform of  $u_s(t)$  does not exist because its defining improper integral does not converge in the absolute sense—a situation we describe as **non-integrable**. The Laplace transform *does* exist for  $u_s(t)$  because it patches the Fourier transform integrand with a weighting function  $w : \mathbb{R} \rightarrow \mathbb{R}$  defined as

$$w(t) = e^{-\sigma t} \quad (1)$$

for  $\sigma \in \mathbb{R}$ . Clearly such a factor may drive the integrand \_\_\_\_\_ for some positive  $\sigma$ . Let’s take the Fourier transform of a function of time  $f$  multiplied by this weightin factor (as a foreshadowing of how the Laplace transform will use it):

$$\mathcal{F}(f(t)w(t)) = \int_{-\infty}^{\infty} f(t)w(t)e^{-j\omega t} dt \quad (\text{FT def.})$$

$$= \int_{-\infty}^{\infty} f(t)e^{-\sigma t}e^{-j\omega t} dt \quad (2a)$$

$$= \int_{-\infty}^{\infty} f(t)e^{-(\sigma+j\omega)t} dt. \quad (2b)$$

<sup>1</sup>See (Rowell and Wormley, 1997, ch. 15) for an introduction that inspires our own.

We see the factor  $\sigma + j\omega$  has emerged. This factor also arises in the Laplace transform, so we make an explicit definition.

**Definition 11 lap.1: Laplace  $s$**

The Laplace  $s \in \mathbb{C}$  (a.k.a. complex frequency) is defined as

$$s = \sigma + j\omega$$

for  $\sigma, \omega \in \mathbb{R}$ .

4 The ubiquity of  $s$  has generated a common term called the \_\_\_\_\_, which is used as an alias for the set of complex numbers  $\mathbb{C}$ , which, when considering its real and imaginary parts to constitute two Cartesian axes (i.e.  $\mathbb{R}^2$ ) charts a plane.

5 Returning to our Fourier transform,

$$\mathcal{F}(f(t)w(t)) = \int_{-\infty}^{\infty} f(t)e^{-st} dt. \quad (3)$$

This is sometimes called the **two-sided Laplace transform**, which is rarely used. However, it is instructive to recognize that potentially, for some region of  $s$ -values in the complex plane, the transform exists. We call this the \_\_\_\_\_ (ROC) of the transform.

6 Now consider what happens if  $f(t) = u_s(t)$ , the unit step that doesn't have a Fourier transform, but the two-sided transform of Eq. 2a yields

$$\begin{aligned} \mathcal{F}(u_s(t)w(t)) &= \int_{-\infty}^{\infty} u_s(t)e^{-\sigma t}e^{-j\omega t} dt \\ &\mathcal{F}(u_s(t)e^{-\sigma t}), \end{aligned}$$

a straightup Fourier transform of  $u_s(t)e^{-\sigma t}$ . Consulting Table ft.1, we see that the transform is

$$\mathcal{F}(u_s(t)e^{-\sigma t}) = \frac{1}{\sigma + j\omega}. \quad (4)$$

So, although  $\mathcal{F}(u_s(t))$  \_\_\_\_\_,  $\mathcal{F}(u_s(t)e^{-\sigma t})$  does. This bodes well for the Laplace transform.



## 11.2 lap.def Laplace transform and its inverse

### The Laplace transform

1 The two-sided definition of the Laplace transform was encountered in [Lec. 11.1 lap.in](#). This is rarely used in engineering analysis, which prefers the following one-sided transform.<sup>2</sup>

#### Definition 11 lap.2: Laplace transform

Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}$  be a function of time  $t$  for which  $f(t) = 0$  for  $t < 0$ . The Laplace transform<sup>3</sup>  $\mathcal{L} : T \rightarrow S$  of  $f$  is defined as<sup>4</sup>

$$(\mathcal{L}f)(s) = \int_0^{\infty} f(t)e^{-st} dt.$$

2 As with the Fourier transform image, it is customary to \_\_\_\_\_ the Laplace transform image; e.g.<sup>5</sup>

$$F(s) = (\mathcal{L}f)(s).$$

3 As with the two-sided Laplace transform, if the transform exists, it will do so for some region of convergence (ROC), a subset of the  $s$ -plane. It is best practice to report a Laplace transform image paired with its ROC.

4 On the imaginary axis ( $\sigma = 0$ ),  $s = j\omega$  and the Laplace transform is

$$(\mathcal{L}f)(s) = \int_0^{\infty} f(t)e^{-j\omega t} dt, \quad (1)$$

which is the one-sided Fourier transform! Therefore, when the Laplace transform exists for a region of convergence that \_\_\_\_\_

<sup>2</sup>We will refer exclusively to the one-sided transform as the Laplace transform and will qualify “two-sided” in the other case.

<sup>3</sup>Here  $T = L^2(0, \infty)$  is the set of square-integrable functions on the positive reals and  $S = H^2(\mathbb{C}_+)$  is a Hardy space with square norm on the (complex) right half-plane (Partington, 2004, p. 7). This highly mathematical notation highlights the fact that the Laplace transform maps a real function of  $t$  to a complex function of  $s$ .

<sup>4</sup>For more detail, see Rowell and Wormley (1997), Dyke (2014), and Mathews and Howell (2012).

<sup>5</sup>Another common notation is  $\mathcal{L}(f(t))$ .

\_\_\_\_\_ , the one-sided Fourier transform also exists and<sup>6</sup>

$$(\mathcal{F}f)(\omega) = (\mathcal{L}f)(s)|_{s \rightarrow j\omega} \quad (2)$$

or, haphazardly using  $F$  to denote both transforms,

$$F(\omega) = F(s)|_{s \rightarrow j\omega}. \quad (3)$$

### Box 11 lap.1 Laplace terminology

The terminology in the literature for the Laplace transform and its inverse, introduced next, is inconsistent. The “Laplace transform” is at once taken to be a function that maps a function of  $t$  to a function of  $s$  and a particular result of that mapping (technically the *image* of the map), which is a complex function of  $s$ . Even we will say things like “the value of the Laplace transform  $F(s)$  at  $s = 2 + j4$ ,” by which we really mean the image of the complex function  $F(s)|_{s \rightarrow 2+j4}$  that was the image of the Laplace transform map of the real function of time  $f(t)$ . You can see why we shorten it.

## The inverse Laplace transform

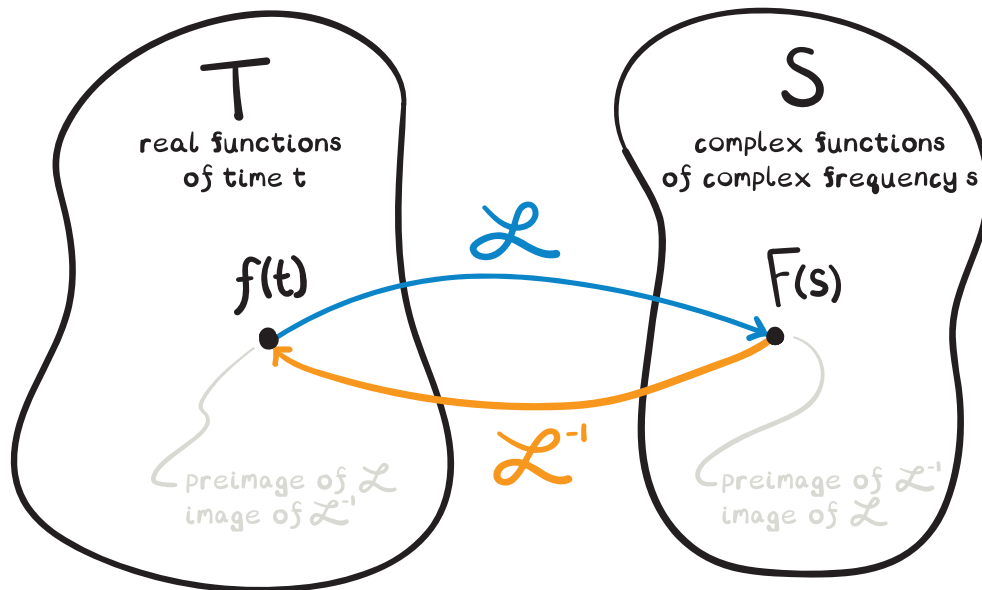
5 As with the Fourier transform, the Laplace transform has an \_\_\_\_\_ .

### Definition 11 lap.3: Inverse Laplace transform

Let  $s \in \mathbb{C}$  be the Laplace  $s$  and  $F(s)$  a Laplace transform image of real function  $f(t)$ . The *inverse Laplace transform*  $\mathcal{L}^{-1} : S \rightarrow T$  is defined as

$$(\mathcal{L}^{-1}F)(t) = \frac{1}{j2\pi} \int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st} ds.$$

<sup>6</sup>The same relation can be shown to hold between the two-sided Fourier and Laplace transforms.

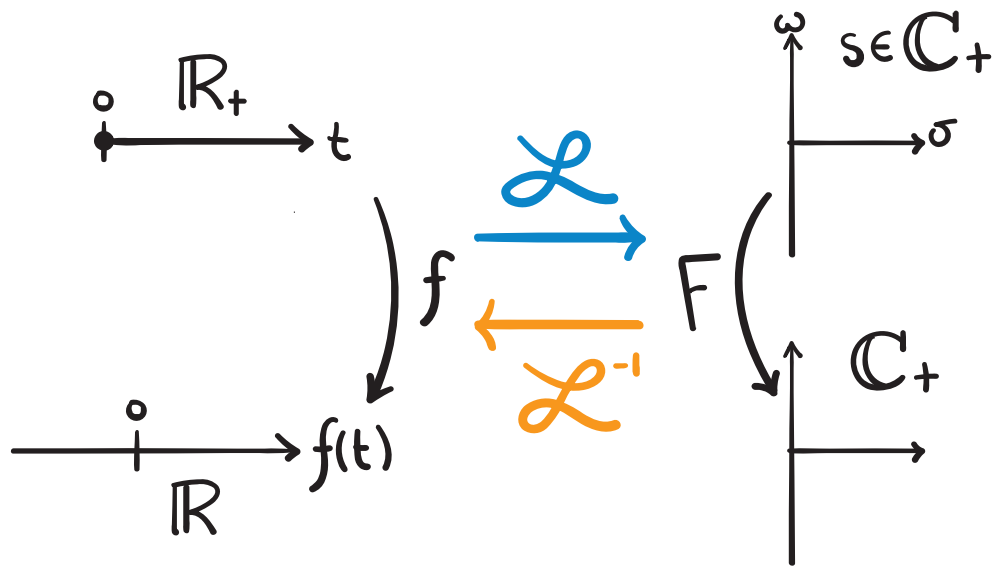


**Figure def.1:** Laplace transform maps  $\mathcal{L}$  and  $\mathcal{L}^{-1}$  on the function spaces T and S.

As is illustrated in Fig. def.1, it can be shown that the inverse Laplace transform image of a Laplace transform image of  $f(t)$  equals  $f(t)$  and vice-versa; i.e.

$$(\mathcal{L}^{-1}\mathcal{L}f)(t) = f(t) \quad \text{and} \\ (\mathcal{L}\mathcal{L}^{-1}F)(s) = F(s).$$

That is, the inverse Laplace transform is a true inverse. Therefore, we call the Laplace transform and its inverse a \_\_\_\_\_.



**Figure def.2:** detail view of Laplace transform maps  $\mathcal{L}$  and  $\mathcal{L}^{-1}$  along with their image functions  $f$  and  $F$ .

6 A detail view of Fig. def.1 is given in Fig. def.2.

### Example 11.2 lap.def-1

Returning to the troublesome unit step  $f(t) = u_s(t)$ , calculate its Laplace transform image  $F(s)$ .

Directly applying the definition,

$$\begin{aligned}
 F(s) &= \int_0^{\infty} f(t)e^{-st} dt \\
 &= \underline{\hspace{2cm}} \\
 &= \frac{-1}{s} e^{-st} \Big|_{t=0}^{\infty} \\
 &= \left( \lim_{t \rightarrow \infty} -e^{-(\sigma+j\omega)t/s} \right) - \frac{-1}{s} e^0 && (s = \sigma + j\omega) \\
 &= 1/s. && (\sigma > 0)
 \end{aligned}$$

• Note that the limit only converges for  $\sigma > 0$ , so the region of convergence

is the right half  $s$ -plane, exclusive of the imaginary axis. This exclusion tells us what we already know, that the Fourier transform  $u_s$  does not exist. However, the Laplace transform *does* exist and is simply  $1/s$ !

7 Both the Laplace transform and especially its inverse are typically calculated with the help of software and tables such as [Table lap.1](#), which includes specific images and important properties. We will first consider these properties in [Lec. 11.3 lap.pr](#), then turn to the use of software and tables in [Lec. 11.4 lap.inv](#) where we focus on the more challenging inverse calculation.

## 11.3 lap.pr Properties of the Laplace transform

1 The Laplace transform has several important properties, several of which follow from the simple fact of its \_\_\_\_\_ definition. We state the properties without proof, but several are easy to show and make good exercises.

### Existence

2 As we have already seen, the Laplace transform exists for more functions than does the Fourier transform. Let  $f : \mathbb{R}_+ \rightarrow \mathbb{R}$  have a finite number of finite-magnitude discontinuities. If there can be found  $M, \alpha \in \mathbb{R}$  such that

$$|f(t)| \leq M e^{\alpha t} \quad \forall t \in \mathbb{R}_+ \quad (1)$$

then the transform exists (converges) for  $\sigma > \alpha$ .

3 Note that this is a \_\_\_\_\_ condition, not necessary. That is, there may be (and are) functions for which a transform exists that do not meet the condition above.

### Linearity

4 The Laplace transform is a \_\_\_\_\_ map. Let  $a, b \in \mathbb{R}$ ;  $f, g \in T$  where  $T$  is a set of functions of nonnegative time  $t$ ; and  $F, G$  the Laplace transform images of  $f, g$ . The following identity holds:

$$\mathcal{L}(af(t) + bg(t)) = aF(s) + bG(s). \quad (2)$$

### Time-shifting

5 Shifting the time-domain function  $f(t)$  in time corresponds to a simple product in the  $s$ -domain Laplace transform image. Let the Laplace transform image of  $f(t)$  be  $F(s)$  and  $\tau \in \mathbb{R}$ . The following identity holds:

$$\mathcal{L}(f(t + \tau)) = e^{s\tau} F(s). \quad (3)$$

### Time-differentiation

6 \_\_\_\_\_ the time-domain function  $f(t)$  with respect to time yields a simple relation in the  $s$ -domain. Let  $F(s)$  be the Laplace transform image of  $f(t)$  and  $f(0)$  the value of  $f$  at  $t = 0$ . The following identity holds:<sup>7</sup>

$$\mathcal{L} \frac{df}{dt} = sF(s) - f(0). \tag{4}$$

### Time-integration

7 Similarly, \_\_\_\_\_ the time-domain function  $f(t)$  with respect to time yields a simple relation in the  $s$ -domain. Let  $F(s)$  be the Laplace transform image of  $f(t)$ . The following identity holds:<sup>8</sup>

$$\mathcal{L} \int_0^t f(\tau) d\tau = \frac{1}{s} F(s). \tag{5}$$

### Convolution

8 The convolution operator  $*$  is defined for real functions of time  $f, g$  by

$$(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau. \tag{6}$$

This too has a simple Laplace transform. Let  $F, G$  be the Laplace transforms of  $f, g$ . The following identity holds:

$$\mathcal{L}(f * g)(t) = F(s)G(s). \tag{7}$$

### Final value theorem

9 The **final value theorem** is a property of the Laplace transform. This theorem allows the computation of \_\_\_\_\_ time-domain steady-state values from the frequency domain, which can be quite convenient when the inverse Laplace transform is elusive. Let  $f(t)$  have

<sup>7</sup>For this reason, it is common for  $s$  to be called the *differentiator*, but this is imprecise and pretty bush league.

<sup>8</sup>For this reason, it is common for  $1/s$  to be called the *integrator*.

transform  $F(s)$  and its time-derivative have an existing transform. If the limit in time exists,

$$\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s). \quad (8)$$

Note that if the steady-state of  $f(t)$  is not a constant (e.g. it is sinusoidal), the limit does not exist.



## 11.4 lap.inv Inverse Laplace transforming

1 The inverse Laplace transform is a \_\_\_\_\_ in the  $s$ -plane, and it can be quite challenging to calculate. Therefore, software and tables such as Table ft.1 are typically applied, instead. In system dynamics, it is common to apply the inverse Laplace transform to a ratio (or products thereof) of polynomials in  $s$  like

$$\frac{a_m s^m + a_{m-1} s^{m-1} + \dots + a_0}{b_n s^n + b_{n-1} s^{n-1} + \dots + b_0} \tag{1}$$

for  $a_i, b_i \in \mathbb{R}$ . However, inverse transforms of general ratios such as these do not appear in the tables. Instead, low-order polynomial ratios *do* appear and have simple inverse Laplace transforms. Suppose we could decompose Eq. 1 into smaller additive terms. Due to the *linearity* property of the inverse Laplace transform, each transform could be calculated separately and consequently summed.

2 The name given to the process of decomposing Eq. 1 into smaller \_\_\_\_\_ terms is called **partial fraction expansion**<sup>9</sup>. It is not particularly difficult, but it is rather tedious. Fortunately, several software tools have been developed for this expansion.

### Inverse transform with a partial fraction expansion in Matlab

3 Matlab’s Symbolic Math toolbox function `partfrac` is quite convenient.

```
help partfrac
```

4 Let’s apply this to an example.

#### Example 11.4 lap.inv-1

What is the inverse Fourier transform image of

$$F(s) = \frac{s^2 + 2s + 2}{s^2 + 6s + 36} \cdot \frac{6}{s + 6} \tag{2}$$

<sup>9</sup> Rowell and Wormley, 1997, App. C.

First, define a symbolic  $s$ .

```
syms s 'complex'
```

Now we can define  $F$ , a symbolic expression for  $F(s)$ .

```
F = (s^2 + 2*s + 2)/(s^2 + 6*s + 36)*6/(s+6);
```

Now all that remains is to apply `partfrac`.

```
F_pf = partfrac(F)
```

```
F_pf =
13/(3*(s + 6)) + ((5*s)/3 - 24)/(s^2 + 6*s + 36)
```

Now consider the Laplace transform table. The first term can easily be inverted:

$$\mathcal{L}^{-1}\left(\frac{13}{3} \cdot \frac{1}{s+6}\right) = \frac{13}{3} \mathcal{L}^{-1} \frac{1}{s+6} \quad (\text{linearity})$$

$$\frac{13}{3} e^{-6t}. \quad (\text{table})$$

The second term, call it  $F_2$ , is not quite as obvious, but the preimage

$$\frac{s-a}{(s-a)^2 + \omega^2} \quad (3)$$

is close. Let's first make the numerator match:

$$\frac{5}{3}s - 24 = \frac{5}{3} \left(s - \frac{72}{5}\right), \quad (4)$$

so  $a_1 = 72/5$ . Now we need the term  $(s-a_1)^2$  in the denominator. Asserting the equality

$$s^2 + 6s + 36 = (s-a_2)^2 + \omega^2$$

$$= s^2 - 2a_2s + a_2^2 + \omega^2.$$

Equating the  $s^0$  coefficients yields  $\omega^2 = 36 - a_2^2$  and equating the  $s$  coefficient yields  $a_2 = -3 \neq a_1 = 72/5$ , so no cigar! What if we “force” the rule by using a new  $a'_1 = a_2$ , which can be achieved by adding a term (and subtracting it elsewhere)? We need  $a'_1 = -3$ , so if we add (and subtract) a term

$$\frac{\frac{5}{3}(a_1 - a'_1)}{(s - a_2)^2 + \omega^2},$$

like

$$F_2 = \frac{\frac{5}{3}(s - a_1)}{(s - a_2)^2 + \omega^2} + \frac{\frac{5}{3}(a_1 - a'_1)}{(s - a_2)^2 + \omega^2} - \frac{\frac{5}{3}(a_1 - a'_1)}{(s - a_2)^2 + \omega^2}$$

we can combine the first two terms to yield

$$F_2 = \frac{\frac{5}{3}(s - a'_1)}{(s - a_2)^2 + \omega^2} - \frac{\frac{5}{3}(a_1 - a'_1)}{(s - a_2)^2 + \omega^2}$$

where we recall that  $a'_1 = a_2$  by construction.

Now the expression is

$$F_2 = \frac{\frac{5}{3}(s - a_2)}{(s - a_2)^2 + \omega^2} - \frac{\frac{5}{3}(a_1 - a_2)}{(s - a_2)^2 + \omega^2}$$

The first term is, by construction, in the Laplace transform table. The second term is close to

$$\frac{\omega}{(s - a)^2 + \omega^2}$$

for which we must make the numerator equal  $\omega$ . Our  $\omega^2 = 36 - a_2^2 = 27$ , so  $\omega = \pm\sqrt{27}$ . The current numerator is

$$\begin{aligned} \frac{5}{3}(a_1 - a_2) &= \frac{5}{3} \left( \frac{72}{5} + 3 \right) \\ &= 29. \end{aligned}$$

So we factor out  $29/\sqrt{27}$  to yield

$$\frac{\frac{29}{\sqrt{27}}\omega}{(s - a_2)^2 + \omega^2}$$

• Returning to  $F_2$ , we have arrived at

$$F_2 = \frac{\frac{5}{3}(s - a_2)}{(s - a_2)^2 + \omega^2} - \frac{\frac{29}{\sqrt{27}}\omega}{(s - a_2)^2 + \omega^2}$$

Now the inverse transform is

$$\begin{aligned}\mathcal{L}^{-1}F_2 &= \frac{5}{3}\mathcal{L}^{-1}\frac{(s - a_2)}{(s - a_2)^2 + \omega^2} - \frac{29}{\sqrt{27}}\mathcal{L}^{-1}\frac{\omega}{(s - a_2)^2 + \omega^2} && \text{(linearity)} \\ &= \frac{5}{3}e^{a_2t} \cos \omega t - \frac{29}{\sqrt{27}}e^{a_2t} \sin \omega t.\end{aligned}$$

Simple! Putting it all together, then,

$$F(s) = \frac{13}{3}e^{-6t} + \frac{5}{3}e^{-3t} \cos(3\sqrt{3}t) - \frac{29}{3\sqrt{3}}e^{-3t} \sin(3\sqrt{3}t).$$

5 You may have noticed that even with Matlab's help with the partial fraction expansion, the inverse Laplace transform was a bit messy. This will motivate you to learn the technique in the next section.

### Just clubbing it with Matlab

6 Sometimes we can just use Matlab (or a similar piece of software) to compute the transform.

7 Matlab's Symbolic Math toolbox function for the inverse Laplace transform is `ilaplace` (and for the Laplace transform, `laplace`).

```
help ilaplace
```

8 Let's apply this to the same example.

### Example 11.4 lap.inv-2

What is the inverse Fourier transform image of

$$F(s) = \frac{s^2 + 2s + 2}{s^2 + 6s + 36} \cdot \frac{6}{s + 6} \quad (5)$$

Use Matlab's `ilaplace`.

First, define a symbolic `s`.

```
syms s 'complex'
```

Now we can define `F`, a symbolic expression for  $F(s)$ .


```
F = (s^2 + 2*s + 2)/(s^2 + 6*s + 36)*6/(s+6);
```

Now all that remains is the apply `ilaplace`.

```
F_pf = ilaplace(F)
```

```
F_pf =
• (13*exp(-6*t))/3 + (5*exp(-3*t))*(cos(3*3^(1/2)*t) -
• ↪ (29*3^(1/2)*sin(3*3^(1/2)*t))/15)/3
```

• This is easily seen to be equivalent to our previous result


$$F(s) = \frac{13}{3}e^{-6t} + \frac{5}{3}e^{-3t} \cos(3\sqrt{3}t) - \frac{29}{3\sqrt{3}}e^{-3t} \sin(3\sqrt{3}t).$$

## 11.5 lap.sol Solving io ODEs with Laplace

1 Laplace transforms provide a convenient method for solving input-output (io) ordinary differential equations (ODEs).

2 Consider a dynamic system described by the \_\_\_\_\_—with  $t$  time,  $y$  the *output*,  $u$  the *input*, constant coefficients  $a_i, b_j$ , order  $n$ , and  $m \leq n$  for  $n \in \mathbb{N}_0$ —as:

$$\begin{aligned} \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = \\ b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u. \end{aligned} \quad (1)$$

Re-written in summation form,

$$\sum_{i=0}^n a_i y^{(i)}(t) = \sum_{j=0}^m b_j u^{(j)}(t), \quad (2)$$

where we use [Lagrange's notation](#) for derivatives, and where, \_\_\_\_\_,  $a_n = 1$ .

3 The Laplace transform  $\mathcal{L}$  of [Eq. 2](#) yields

$$\mathcal{L} \sum_{i=0}^n a_i y^{(i)}(t) = \mathcal{L} \sum_{j=0}^m b_j u^{(j)}(t) \quad \Rightarrow \quad (3a)$$

$$\sum_{i=0}^n a_i \mathcal{L} \left( y^{(i)}(t) \right) = \sum_{j=0}^m b_j \mathcal{L} \left( u^{(j)}(t) \right). \quad (\text{linearity})$$

In the next move, we recursively apply the \_\_\_\_\_ property to yield the following

$$\sum_{i=0}^n a_i \left( s^i Y(s) + \underbrace{\sum_{k=1}^i s^{i-k} y^{(k-1)}(0)}_{I_i(s)} \right) = \sum_{j=0}^m b_j s^j U(s), \quad (4)$$

where terms in  $I_i(s)$  arise from the \_\_\_\_\_. Splitting the left outer sum and solving for  $Y(s)$ ,

$$\sum_{i=0}^n a_i s^i Y(s) + \sum_{i=0}^n a_i I_i(s) = \sum_{j=0}^m b_j s^j U(s) \Rightarrow \quad (5a)$$

$$\sum_{i=0}^n a_i s^i Y(s) = \sum_{j=0}^m b_j s^j U(s) - \sum_{i=0}^n a_i I_i(s) \Rightarrow \quad (5b)$$

$$Y(s) \sum_{i=0}^n a_i s^i = U(s) \sum_{j=0}^m b_j s^j - \sum_{i=0}^n a_i I_i(s) \Rightarrow \quad (5c)$$

$$Y(s) = \underbrace{\frac{\sum_{j=0}^m b_j s^j}{\sum_{i=0}^n a_i s^i}}_{Y_{fo}(s)} U(s) + \underbrace{\frac{-\sum_{i=0}^n a_i I_i(s)}{\sum_{i=0}^n a_i s^i}}_{Y_{fr}(s)}. \quad (5d)$$

4 So we have derived the \_\_\_\_\_  $Y(s)$  in terms of the **forced** and **free** responses (still in the  $s$ -domain, of course)! For a solution in the time-domain, we must inverse Laplace transform:

$$y(t) = \underbrace{(\mathcal{L}^{-1} Y_{fo})(t)}_{y_{fo}(t)} + \underbrace{(\mathcal{L}^{-1} Y_{fr})(t)}_{y_{fr}(t)}. \quad (6)$$

This is an important result!

### Example 11.5 lap.sol-1

Consider a system with step input  $u(t) = 7u_s(t)$ , output  $y(t)$ , and io ODE

$$\ddot{y} + 2\dot{y} + y = 2u. \quad (7)$$

Solve for the *forced* response  $y_{fo}(t)$  with Laplace transforms.

5 From Eq. 6,

$$\begin{aligned} y_{fo}(t) &= (\mathcal{L}^{-1} Y_{fo})(t) \\ &= \mathcal{L}^{-1} \left( \frac{\sum_{j=0}^m b_j s^j}{\sum_{i=0}^n a_i s^i} U(s) \right) \end{aligned} \quad (\text{Eq. 5d})$$

$$= \mathcal{L}^{-1} \left( \frac{2}{s^2 + 2s + 1} U(s) \right). \quad (\text{Eq. 7})$$



• We can \_\_\_\_\_  $u(t)$  for  $U(s)$ :

$$\begin{aligned} U(s) &= (\mathcal{L}u)(s) \\ &= 7(\mathcal{L}u_s)(s) \\ &= \frac{7}{s}, \end{aligned}$$

where the last equality follows from a transform easily found in [Table lap.1](#).

6 Returning to the time response \_\_\_\_\_,

$$\begin{aligned} y_{fo}(t) &= \mathcal{L}^{-1} \left( \frac{2}{s^2 + 2s + 1} U(s) \right) \\ &= \mathcal{L}^{-1} \left( \frac{2}{s^2 + 2s + 1} \cdot \frac{7}{s} \right). \end{aligned}$$

7 We can use Matlab's Symbolic Math toolbox function `partfrac` to perform the partial fraction expansion.

```
syms s 'complex'
Y = 2/(s^2 + 2*s + 1)*7/s;
Y_pf = partfrac(Y)
```

Y\_pf =

```
14/s - 14/(s + 1)^2 - 14/(s + 1)
```

Or, a little nicer to look at,

$$Y(s) = 14 \left( \frac{1}{s} - \frac{1}{(s+1)^2} - \frac{1}{s+1} \right).$$

Substituting this into our solution,

$$\begin{aligned} y_{fo}(t) &= 14\mathcal{L}^{-1} \left( \frac{1}{s} - \frac{1}{(s+1)^2} - \frac{1}{s+1} \right) && \text{(linearity)} \\ &= 14 \left( \mathcal{L}^{-1} \frac{1}{s} - \mathcal{L}^{-1} \frac{1}{(s+1)^2} - \mathcal{L}^{-1} \frac{1}{s+1} \right) \\ &= 14 (u_s(t) - te^{-t} - e^{-t}) && \text{(Table lap.1)} \\ &= 14 (u_s(t) - (t+1)e^{-t}). \end{aligned}$$

So the forced response starts at 0 and decays \_\_\_\_\_ to a steady 14.

## **11.6 lap.exe Exercises for Chapter 11** **lap**

1 In [Lec. 03.7 ss.ss2tf2io](#), we briefly introduced the transfer function, a very important dynamic system representation to which we now turn our attention. We repeat the definition here.

2 Let a system have an input  $u$  and an output  $y$ . Let the Laplace transform of each be denoted  $U$  and  $Y$ , both functions of complex Laplace transform variable  $s$ . A **transfer function**  $H$  is defined as the ratio of the Laplace transform of the output over the input:

$$H(s) = \frac{Y(s)}{U(s)}. \quad (1)$$

3 The transfer function is exceedingly useful in many types of analysis. One of its most powerful aspects is that it gives us access to thinking about systems as operating on an input  $u$  and yielding an output  $y$ .

4 As we learned in [Lec. 03.7 ss.ss2tf2io](#), one can easily convert a state-space model to a (matrix) transfer function model with the formula

$$H(s) = C(sI - A)^{-1}B + D. \quad (2)$$

We also learned that a transfer function and an io ODE are related via the Laplace transform. The similarities are rather easy to spot, so io ODEs and transfer functions can be converted to each other via inspection.

## 12.1 tf.zp Poles and zeros

1 Two important types of objects defined from a transfer function  $H$  can be used to characterize a system's behavior: **poles** and **zeros**.

### Definition 12 tf.1: poles

Let a system have transfer function  $H$ . Its *poles* are values of  $s$  for which

$$|H(s)| \rightarrow \infty.$$

2 A transfer function written as a ratio has poles wherever the denominator is zero; that is,  $s$  for which<sup>1</sup>

### Definition 12 tf.2: zeros

Let a system have transfer function  $H$ . Its *zeros* are values of  $s$  for which

$$|H(s)| \rightarrow 0.$$

3 A transfer function written as a ratio has zeros wherever the numerator is zero; that is,  $s$  for which<sup>2</sup>

4 Given a transfer function  $H$  with  $n$  poles  $p_i$  and  $v$  zeros  $z_j$ , we can write, for  $K \in \mathbb{R}$ ,

<sup>1</sup>It is common to use this as the definition of a pole, which allows us to talk of "pole-zero cancellation." Occasionally we will use this terminology.

<sup>2</sup>It is common to use this as the definition of a zero, which allows us to talk of "pole-zero cancellation." Occasionally we will use this terminology.

$$H(s) = K \frac{\prod_{j=1}^{\nu} (s - z_j)}{\prod_{i=1}^n (s - p_i)}$$

- 5 Poles and zeros can define a single-input, single-output (SISO) system's dynamic model, within a constant.
- 6 Recall that, even for multiple-input, multiple-output (MIMO) state-space models, the denominator of every transfer function is the corresponding system's *characteristic equation*—the roots of which dominate the system's response and are equal to its *eigenvalues*. It is now time to observe a crucial identity.

**Corollary 12 tf.3: poles = eigenvalues = char. eq. roots**

A system's *poles* equal its *eigenvalues* equal its *characteristic equation roots*.

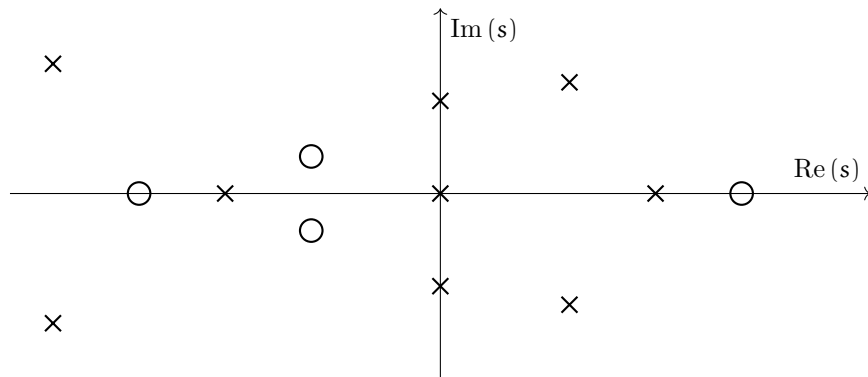
- 7 Therefore, everything we know about a system's eigenvalues and characteristic equation roots is true for a system's poles. This includes that they characterize a system's response (especially its free response) and stability.

**Pole-zero plots and stability**

8 The complex-valued poles and zeros dominate system behavior via their values and value-relationships. Often, we construct a **pole-zero plot**—a plot in the complex plane of a system's poles and zeros—such as that of Fig. zp.1.

9 From our identification of poles with eigenvalues and roots of the characteristic equation, we can recognize that each pole contributes an exponential response that oscillates if it is complex. There are three stability contribution possibilities for each pole  $p_i$ :

- $\text{Re}(p_i) < 0$ : a stable, decaying contribution;
- $\text{Re}(p_i) = 0$ : a marginally stable, neither decaying nor growing contribution; and
- $\text{Re}(p_i) > 0$ : an unstable, growing contribution.



**Figure zp.1:** a pole-zero plot for a system with nine poles and four zeros. In this example, six of the poles are complex-conjugate pairs and three are real. Three are in the right half-plane, making the system unstable. One zero is in the right half-plane, making the system “minimum phase.”

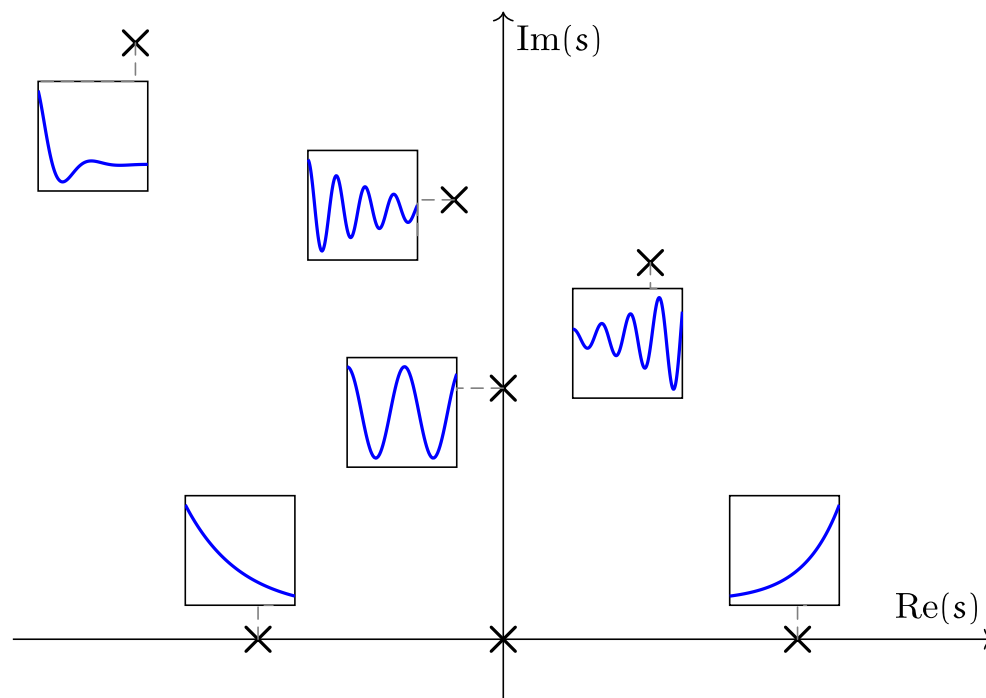
This is explored graphically in [Fig. zp.2](#).

**10** Of course, we must not forget that a *system's* stability is spoiled with a single unstable pole.

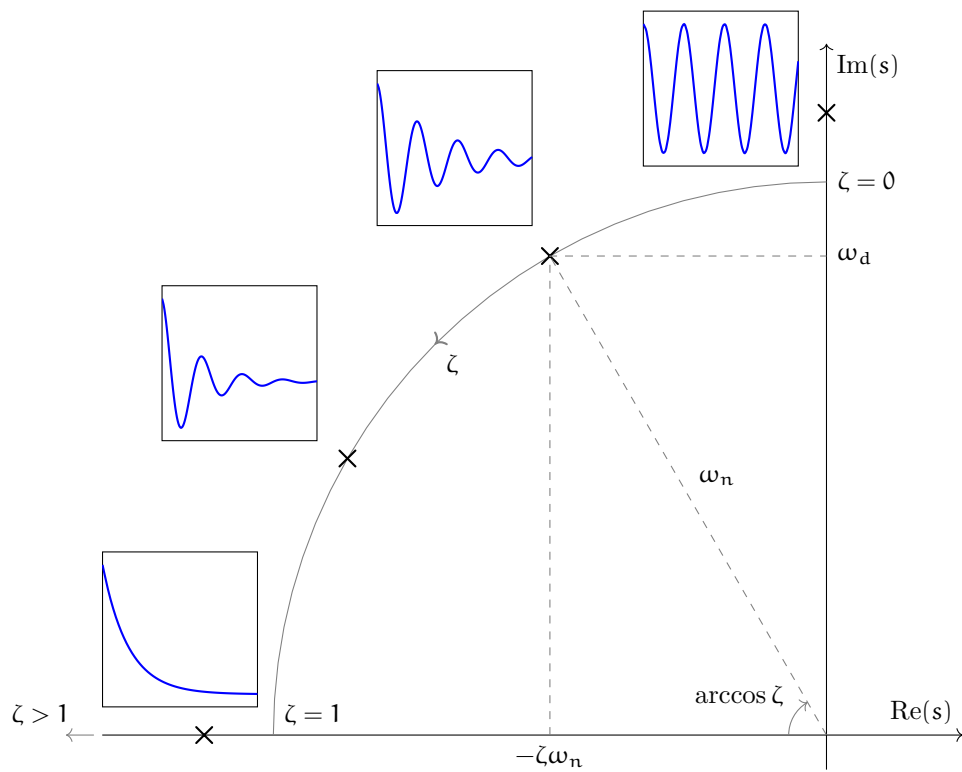
**11** It can be shown that complex poles and zeros always arise as conjugate pairs. A consequence of this is that the pole-zero plot is always **symmetric about the real axis**.

### Second-order systems

**12** Second-order response is characterized by a damping ratio  $\zeta$  and natural frequency  $\omega_n$ . These parameters have clear complex-plane “geometric” interpretations, as shown in [Fig. zp.3](#). Pole locations are interpreted geometrically in accordance with their relation to rays of constant damping from the origin and circles of constant natural frequency, centered about the origin.



**Figure zp.2:** free response contributions from poles at different locations. Complex poles contribute oscillating free responses, whereas real poles do not. Left half-plane poles contribute stable responses that decay. Right half-plane poles contribute unstable responses that grow. Imaginary-axis poles contribute marginal stability.



**Figure zp.3:** second-order free response contributions from poles at different locations, characterized by the damping ratio  $\zeta$  and natural frequency  $\omega_n$ . Constant damping occurs along rays from the origin. Constant natural frequency occurs along arcs of constant radius, centered at the origin.



## 12.2 tf.tfmat Exploring transfer functions in Matlab

Matlab includes several nice functions for working with transfer functions. We explore some here.

### The `tf` command and its friends

The `tf` command allows us to create LTI transfer function objects (which we'll abbreviate as "tf objects") that are recognized by `lsim`, `step`, and `initial`.

Consider the transfer function

$$H(s) = \frac{s + 1}{s^3 + 3s^2 + 7s + 1}. \quad (1)$$

We can make a Matlab model as follows.

```
sys = tf([1,1],[1,3,7,1])
```

```
sys =
```

```
      s + 1
-----
s^3 + 3 s^2 + 7 s + 1
```

```
Continuous-time transfer function.
```

Alternatively, we could define `s` as a transfer function model itself.

```
s = tf([1,0],[1]); % tf is 1*s+0/1 = s
(s+1)/(s^3+3*s^2+7*s+1)
```

```
ans =
```

```
      s + 1
-----
s^3 + 3 s^2 + 7 s + 1
```

```
Continuous-time transfer function.
```

### Algebraic operations with `tf`s

Say we have two transfer functions  $G(s)$  and  $H(s)$  (already defined as `sys`). We might want to concatenate them. The idea is that we might take the output of  $G(s)$  and use that as the input to  $H(s)$ . In this case, the transfer function from the input of  $G(s)$  to the output of  $H(s)$  is just the multiplication

$$G(s)H(s). \quad (2)$$

```
G = 1/(s+2); % or tf([1],[1,2])
G*sys
```

```
ans =
```

```

              s + 1
-----
s^4 + 5 s^3 + 13 s^2 + 15 s + 2
```

```
Continuous-time transfer function.
```

Note that we have seen that Matlab handles addition and multiplication of scalars and `tf`s as well as the products of `tf`s. (It will also handle division.)

### State-space models to `tf` models.

Consider the state-space model with standard matrices as shown below.

```
A = [-2,0;0,-3];
B = [1;1];
C = [1,0;1,1;0,1];
D = [0;0;1];
```

We can create a `ss` model as usual.

```
sys_ss = ss(A,B,C,D);
```

First, let's form a transfer function symbolically

We know the transfer function matrix is given by

$$C(sI - A)^{-1}B + D. \quad (3)$$

```
syms S
sys_tf_s = C*inv(S*eye(size(A)) - A)*B + D
```

```
sys_tf_s =

          1/(S + 2)
1/(S + 2) + 1/(S + 3)
          1/(S + 3) + 1
```

This gave us three symbolic transfer functions in a  $3 \times 1$  matrix, the first being that for the input to the first output, the second for the input to the second output, etc.

Or we can convert the *ss* model to a *tf* model

We can actually simply pass the *ss* model to the *tf* function.

```
sys_tf = tf(sys_ss)
```

```
sys_tf =

From input to output...
          1
1:  -----
     s + 2

          2 s + 5
2:  -----
     s^2 + 5 s + 6

          s + 4
3:  -----
          s + 3
```

```
Continuous-time transfer function.
```

Note that the function `ss2tf` has a serious bug and should not be trusted.

### Poles, zeros, and stability

Let's take a look at the poles and zeros of `sys`.

```
p_sys = pole(sys)
z_sys = zero(sys)
```

```
p_sys =
-1.4239 + 2.1305i
-1.4239 - 2.1305i
-0.1523 + 0.0000i

z_sys =
-1
```

Stability can be evaluated from `p_sys`. The system is *stable* because the real parts of all poles are negative.

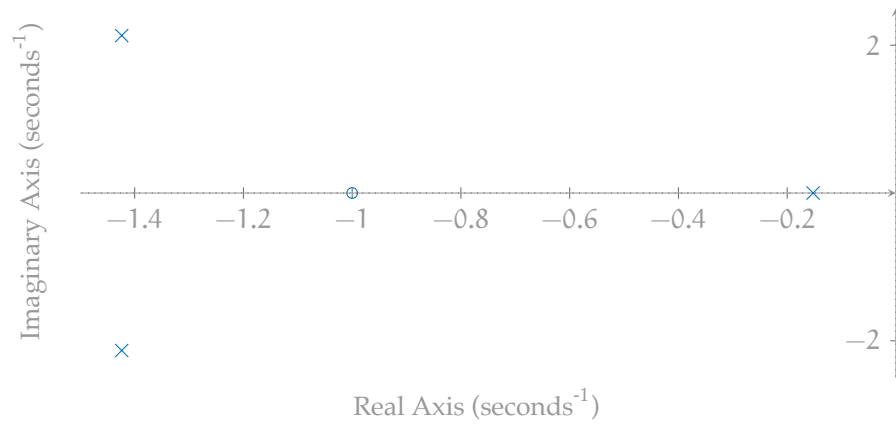
Let's take a look at the pole-zero map.

```
figure;
pzmap(sys)
```

The resulting figure is shown in [Fig. tfmat.1](#).

### Simulating with `tf`s

All the simulation functions we've used for `ss` models (`lsim`, `step`, `impulse`, `initial`) will also work for `tf` models. Let's try a `impulse` response on our original `sys` transfer function model.



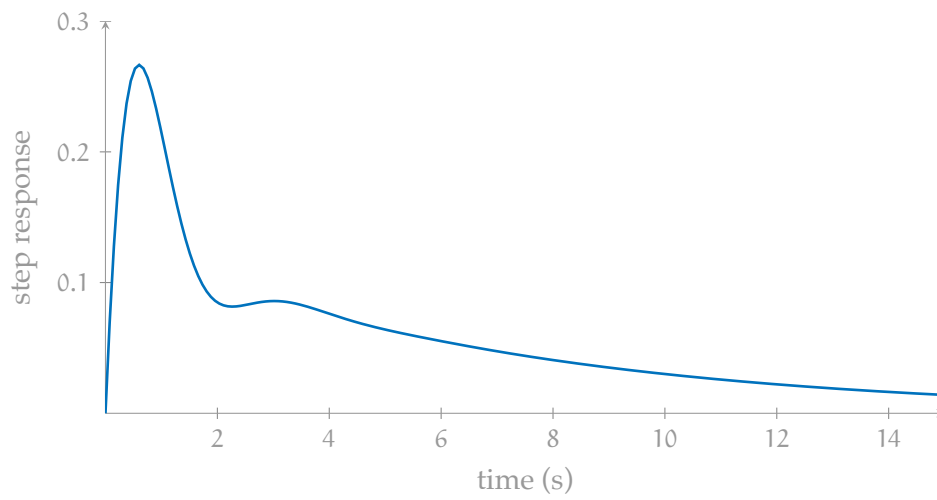
**Figure tfmat.1:** the pole-zero map.

```
t = linspace(0,15,200);  
y = impulse(sys,t);
```

Plot.

```
figure  
plot(t,y);  
xlabel('time (s)')  
ylabel('step response')
```

The resulting figure is shown in [Fig. tfmat.2](#).



**Figure fformat.2:** the impulse response.

## 12.3 tf.zpk ZPK transfer functions in Matlab

Consider the transfer function:

$$H(s) = \frac{2s + 1}{s^2 + 7s + 12} \quad (1)$$

$$= 2 \frac{s + 1/2}{(s + 3)(s + 4)}. \quad (2)$$

In the second equality, we have factored the polynomials and expressed them in terms of poles  $p_i$  and zeros  $z_i$  with terms  $(s - p_i)$  and  $(s - z_i)$ . Note the gain factor 2 that emerges in this form.

Both forms are useful. In the former, two polynomials in  $s$  define the transfer function; in the latter, a list of zeros, poles, and a gain constant define the transfer function.

In Matlab, there are two corresponding manners of defining a transfer function. We demonstrate the first, already familiar, method using the `tf` command, which takes polynomial coefficients, as follows.

```
H_tf = tf([2,1],[1,7,12])
```

```
H_tf =
```

```
      2 s + 1
-----
s^2 + 7 s + 12
```

```
Continuous-time transfer function.
```

Alternatively, we can define the transfer function model with the `zpk` command using the zeros, poles, and gain constant.

```
H_zpk = zpk([-1/2],[-3,-4],2)
```

```
H_zpk =
```

$$\frac{2 (s+0.5)}{(s+3) (s+4)}$$

Continuous-time zero/pole/gain model.

This zpk model will work with all the usual functions tf models do. However, if you'd like to convert zpk to tf, simply use tf as follows.

```
tf(H_zpk)
```

```
ans =
```

$$\frac{2 s + 1}{s^2 + 7 s + 12}$$

Continuous-time transfer function.

Alternatively, we can convert a tf model to a zpk model.

```
zpk(H_tf)
```

```
ans =
```

$$\frac{2 (s+0.5)}{(s+4) (s+3)}$$

Continuous-time zero/pole/gain model.



## 12.4 tf.exe Exercises for Chapter 12 tf

### Exercise 12.1 scallywag

Use a computer to solve this problem. Consider the transfer function

$$H(s) = \frac{10(s+3)}{(s+2)(s^2+8s+41)}.$$

- What are poles and zeros of  $H$ ?
- Comment on the stability of the system described by  $H$  (justify your comment).
- Construct a pole-zero plot.
- Use a function like the Python control package function `step_response` to simulate the unit step response of the system and plot it for  $t \in [0, 3]$  seconds.

### Exercise 12.2 swashbuckling

Consider a system with linear state-space model matrices

$$A = \begin{bmatrix} -1 & 4 \\ 0 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (1a)$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}. \quad (1b)$$

- Derive the transfer function  $H(s)$  for the system. Express it as a single ratio in  $s$ .
- What are the poles and zeros?
- Compare the poles to the eigenvalues of  $A$ .
- Draw or sketch a pole-zero plot.
- With reference to the pole-zero plot, comment on the stability and transient free response characteristics of the system.
- Use the inverse Laplace transform  $\mathcal{L}^{-1}$  to find the system's forced response  $y(t)$  to step input  $u(t) = 9u_s(t)$ .

\_\_\_\_\_/ 25 p.

\_\_\_\_\_/ 30 p.

**Exercise 12.3 boris**

Consider a mass-spring-damper system with mass  $m$ , spring constant  $k$ , and damping coefficient  $B$  with the I/O ODE

$$\ddot{y} + \frac{B}{m}\dot{y} + \frac{k}{m}y = \frac{1}{m}u$$

for input force  $u(t) = F_s(t)$  and output position  $y(t) = x_m(t)$ .

- Find the corresponding transfer function  $H(s) = Y(s)/U(s)$ .
- Find the natural frequency and damping ratio in terms of system parameters  $m$ ,  $k$ , and  $B$ .
- What are poles and zeros of  $H$  in terms of system parameters  $m$ ,  $k$ , and  $B$ ?
- For system parameters  $m = 10$  kg,  $k = 1 \cdot 10^5$  N/m, and  $B = 500$  N·s/m, construct a pole-zero plot.
- Comment on the stability of the system described by  $H$ . Are there any values of system parameters  $m$ ,  $k$ , and  $B$  for which the system is marginally stable or unstable?
- For system parameters  $m = 10$  kg,  $k = 1 \cdot 10^5$  N/m, and  $B = 500$  N·s/m, use a function like the Python control package function `step_response()` to simulate the unit step response of the system and plot it for  $t \in [0, 0.3]$  s.

**13 imp**

---

## 13.1 imp.ip Input impedance and admittance

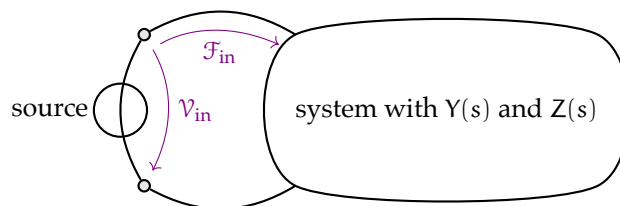
- 1 We now introduce a generalization of the familiar impedance and admittance of electrical circuit analysis, in which system behavior can be expressed algebraically instead of differentially. We begin with generalized input impedance.
- 2 Consider a system with a source, as shown in Fig. ip.1. The source can be either an across- or a through-variable source. The ideal source specifies either  $\mathcal{V}_{in}$  or  $\mathcal{F}_{in}$ , and the other variable depends on the system.
- 3 Let a source variables have Laplace transforms  $\mathcal{V}_{in}(s)$  and  $\mathcal{F}_{in}(s)$ . We define the system's **input impedance**  $Z$  and **input admittance**  $Y$  to be the Laplace-domain ratios

$$Z(s) = \frac{\mathcal{V}_{in}(s)}{\mathcal{F}_{in}(s)} \quad \text{and} \quad Y(s) = \frac{\mathcal{F}_{in}(s)}{\mathcal{V}_{in}(s)}. \quad (1)$$

Clearly,



Both  $Z$  and  $Y$  can be considered **transfer functions**: for a through-variable source  $\mathcal{F}_{in}$ , the impedance  $Z$  is the transfer function to across-variable  $\mathcal{V}_{in}$ ; for an across-variable source  $\mathcal{V}_{in}$ , the admittance  $Y$  is the transfer function to through-variable  $\mathcal{F}_{in}$ . Often, however, we use the more common impedance  $Z$  to characterize systems with either type of source.



**Figure ip.1:**

4 Note that  $Z$  and  $Y$  are **system properties**, not properties of the source. An impedance or admittance can characterize a system of interconnected elements, or a system of a single element, as the next section explores.

### Impedance of ideal passive elements

5 The impedance and admittance of a single, ideal, one-port element is defined from the Laplace transform of its elemental equation.

**Generalized capacitors** A **generalized capacitor** has elemental equation

$$\frac{d\mathcal{V}_C(t)}{dt} = \frac{1}{C}\mathcal{F}_C(t), \quad (2)$$

the Laplace transform of which is

$$s\mathcal{V}_C(s) = \frac{1}{C}\mathcal{F}_C(s), \quad (3)$$

which can be solved for impedance  $Z_C = \mathcal{V}_C/\mathcal{F}_C$  and admittance  $Y_C = \mathcal{F}_C/\mathcal{V}_C$ :

**Generalized inductors** A **generalized inductor** has elemental equation

$$\frac{d\mathcal{F}_L(t)}{dt} = \frac{1}{L}\mathcal{V}_L(t), \quad (4)$$

the Laplace transform of which is

$$s\mathcal{F}_L(s) = \frac{1}{L}\mathcal{V}_L(s), \quad (5)$$

which can be solved for impedance  $Z_L = \mathcal{V}_L/\mathcal{F}_L$  and admittance  $Y_L = \mathcal{F}_L/\mathcal{V}_L$ :

**Generalized resistors** A **generalized resistor** has elemental equation

$$\mathcal{V}_R(t) = \mathcal{F}_R(t)R, \tag{6}$$

the Laplace transform of which is

$$\mathcal{V}_R(s) = \mathcal{F}_R(s)R, \tag{7}$$

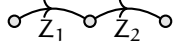
which can be solved for impedance  $Z_R = \mathcal{V}_R/\mathcal{F}_R$  and admittance  $Y_R = \mathcal{F}_R/\mathcal{V}_R$ :




6 For a summary of the impedance of one-port elements, see [Table els.1](#).

**Impedance of interconnected elements**

7 As with electrical circuits, impedances of linear graphs of interconnected elements can be combined in two primary ways: in parallel or in series.

8 Elements sharing the same through-variable are said to be in **series** connection. N elements connected in series  ... have equivalent impedance Z and admittance Y:

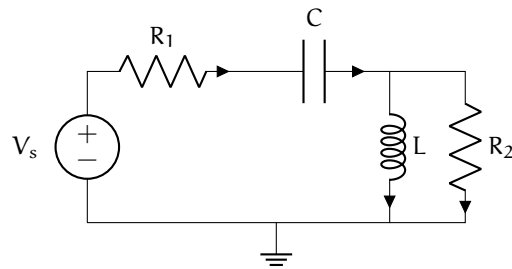
$$Z(s) = \sum_{i=1}^N Z_i(s) \quad \text{and} \quad Y(s) = 1 / \sum_{i=1}^N 1/Y_i(s) \tag{8}$$

9 Conversely, elements sharing the same across-variable are said to be in **parallel** connection. N elements connected in parallel  ... have equivalent impedance Z and admittance Y:

$$Z(s) = 1 / \sum_{i=1}^N 1/Z_i(s) \quad \text{and} \quad Y(s) = \sum_{i=1}^N Y_i(s). \tag{9}$$

**Example 13.1 imp.ip-1**

For the circuit shown, find the input impedance.



re:  
input  
impedance  
of a  
simple  
circuit

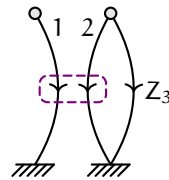
## 13.2 imp.2port Impedance with two-port elements

- 1 The two types of energy transducing elements, **transformers** and **gyrators**, “reflect” or “transmit” impedance through themselves, such that they are “felt” on the other side.
- 2 For a **transformer**, the elemental equations are

$$v_2(t) = v_1(t)/TF \quad \text{and} \quad \mathcal{F}_2(t) = -TF\mathcal{F}_1(t), \tag{1}$$

the Laplace transforms of which are

$$v_2(s) = v_1(s)/TF \quad \text{and} \quad \mathcal{F}_2(s) = -TF\mathcal{F}_1(s). \tag{2}$$



**Figure 2port.1:**

- 3 If, on the 2-side, the input impedance is  $Z_3$ , as in Fig. 2port.1, the equations of Eq. 2 are subject to the continuity and compatibility equations

$$v_2 = v_3 \quad \text{and} \quad \mathcal{F}_2 = -\mathcal{F}_3. \tag{3}$$

Substituting these into Eq. 2 and solving for  $v_1$  and  $\mathcal{F}_1$ ,

$$v_1 = TFv_3 \quad \text{and} \quad \mathcal{F}_1 = \mathcal{F}_3/TF. \tag{4}$$

The elemental equation for element 3 is  $v_3 = \mathcal{F}_3 Z_3$ , which can be substituted into the through-variable equation to yield



- 4 Working our way back from  $v_3$  to  $v_1$ , we apply the compatibility equation  $v_2 = v_3$  and the elemental equation  $v_2 = v_1/TF$ , as follows:

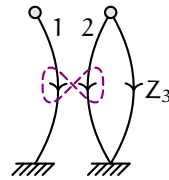




Solving for the **effective input impedance**  $Z_1$ ,

$$Z_1 \equiv \frac{\mathcal{V}_1(s)}{\mathcal{F}_1(s)} \tag{5}$$

$$= TF^2 Z_3. \tag{6}$$



**Figure 2port.2:**

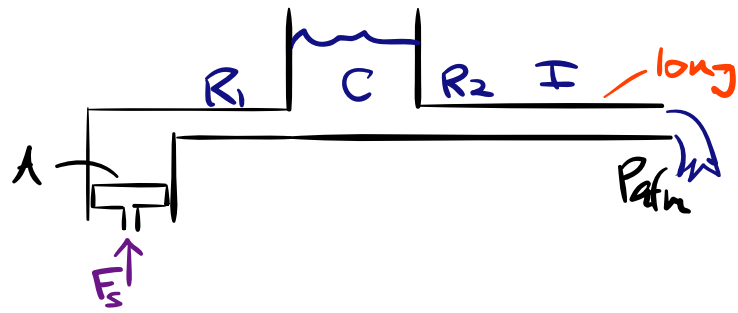
5 For a **gyrator** with gyrator modulus  $GY$ , in the configuration shown in Fig. 2port.2, a similar derivation yields the **effective input impedance**  $Z_1$ ,

$$Z_1 = GY^2/Z_3. \tag{7}$$

**Example 13.2 imp.2port-1**

- ! Draw a linear graph of the fluid system. What is the input impedance for
- an input force to the piston?

re:  
input impedance of fluid system with transducer



## 13.3 `imp.tf` Transfer functions via impedance

- 1 Now the true power of impedance-based modeling is revealed: we can skip a time-domain model (e.g. state-space or io differential equation) and derive a transfer-function model, directly! Before we do, however, let's be sure to recall that a transfer-function model concerns itself with the **forced response** of a system, ignoring the free response. If we care to consider the free response, we can convert the transfer function model to an io differential equation and solve it.
- 2 There are two primary ways impedance-based modeling is used to derive transfer functions. The first and most general is described, here. The second is a shortcut most useful for relatively simple systems; it is described in [Lec. 13.6 `imp.divide`](#).
- 3 In what follows, it is important to recognize that, in the Laplace-domain, every elemental equation is just<sup>1</sup>

$$v = \mathcal{F}Z, \quad (1)$$

where the across-variable, through-variable, and impedance are all element-specific.

- 4 This algorithm is very similar to that for state-space models from linear graph models, presented in [Lec. 03.4 `ss.nt2ss`](#). In the following, we consider a connected graph with  $B$  branches, of which  $S$  are sources (split between through-variable sources  $S_T$  and across  $S_A$ ). There are  $2B - S$  unknown across- and through-variables, so that's how many equations we need. We have  $B - S$  elemental equations and for the rest we will write continuity and compatibility equations.  $N$  is the number of nodes.

1. Derive  $2B - S$  independent Laplace-domain, algebraic equations from Laplace-domain elemental, continuity, and compatibility equations.
  - a) Draw a **normal tree**.

---

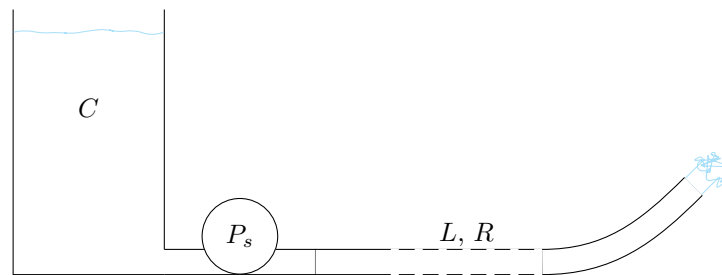
<sup>1</sup>In electronics, this is sometimes called "generalized Ohm's law."

- b) Write a Laplace-domain **elemental equation** for each passive element.<sup>2</sup>
  - c) Write a **continuity equation** for each passive branch by drawing a contour intersecting that and no other branch.<sup>3</sup>
  - d) Write a **compatibility equation** for each passive link by temporarily “including” it in the tree and finding the compatibility equation for the resulting loop.<sup>4</sup>
2. Solve the *algebraic* system of  $2B$  equations and  $2B$  unknowns for outputs in terms of inputs, only. Sometimes, solving for *all* unknowns via the usual methods is easier than trying to cherry-pick the desired outputs.
  3. The solution for each output  $Y_i$  depends on zero or more inputs  $U_j$ . To solve for the transfer function  $Y_i/U_j$ , set  $U_k = 0$  for all  $k \neq j$ , then divide both sides of the equation by  $U_j$ .

### Example 13.3 imp.tf-1

re:  
firehose

For the schematic of a fire hose connected to a fire truck's reservoir  $C$  via pump input  $P_s$ , use impedance methods to find the transfer function from  $P_s$  to the velocity of the spray. Assume the nozzle's cross-sectional area is  $A$ .



<sup>2</sup>There will be  $B - S$  elemental equations.

<sup>3</sup>There will be  $N - 1 - S_A$  independent continuity equations.

<sup>4</sup>There will be  $B - N + 1 - S_T$  independent compatibility equations.



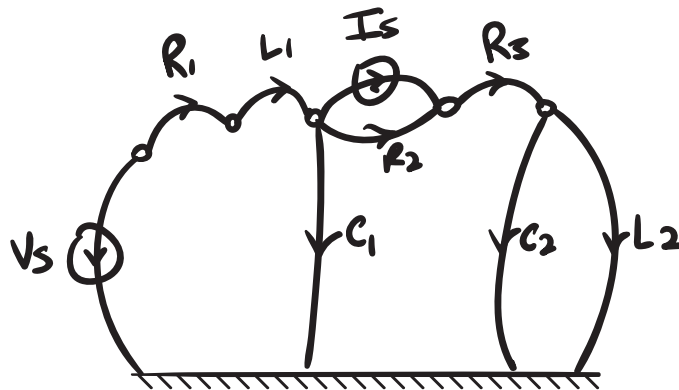


## 13.4 imp.examat Impedance modeling example in Matlab

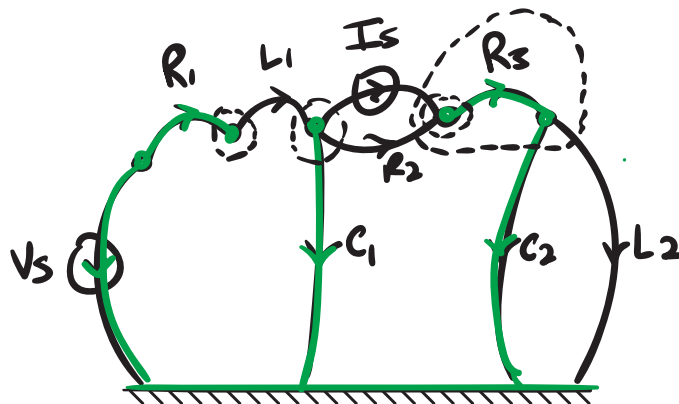
### Example 13.4 imp.examat-1

re:  
Impedance modeling with Matlab

- 1 Consider the linear graph of an electronic system, below. Use impedance methods to derive the transfer functions from inputs  $V_S$  and  $I_S$  to outputs  $v_{C_2}$  and  $i_{R_1}$ .



- 2 The normal tree is shown, below, along with contours to be used for continuity equations.



3 We switch over to Matlab for the remainder of the solution.

Let's define the required symbolic variables.

```
syms s VS IS ...
vC1 iC1 vC2 iC2 vL1 iL1 vL2 iL2 ...
vR1 iR1 vR2 iR2 vR3 iR3 ...
zC1 zC2 zL1 zL2 zR1 zR2 zR3 ...
C1 C2 L1 L2 R1 R2 R3
```

We also specify the unknown variables (two for each passive element), output variables, and input variables.

```
unknowns = [...
vC1 iC1 vC2 iC2 vL1 iL1 vL2 iL2 ...
vR1 iR1 vR2 iR2 vR3 iR3 ...
];
out_i = [3,10]; % output indices
in = [VS;IS]; % input variables
```

Now let's define our elemental, continuity, and compatibility equations.

```
elemental = [...
vC1 == iC1*zC1,...
vC2 == iC2*zC2,...
vL1 == iL1*zL1,...
vL2 == iL2*zL2,...
vR1 == iR1*zR1,...
vR2 == iR2*zR2,...
vR3 == iR3*zR3 ...
];
continuity = [...
iC1 == iL1 - IS - iR2,...
iR1 == iL1,...
iC2 == IS + iR2,...
iR3 == IS + iR2 ...
];
compatibility = [
vL1 == -vR1 + VS - vC1,...
vL2 == vC2,...
```



```
vR2 == vC1 - vC2 - vR3...
];
```

These form a linear system of  $2 \times 7 = 14$  unknowns and 14 equations. Such systems can be defined in matrix form as  $M \cdot \text{unknowns} == b$ , where  $M$  are the coefficients of the unknowns,  $\text{unknowns}$  is the vector of unknowns, and  $b$  is the vector of terms that include the inputs. Matlab has the function `equationsToMatrix` for specifying the matrix form from a list of equations.

```
[M,b] = equationsToMatrix(...
 [elemental,continuity,compatibility],... % eq's
 unknowns... % unknown variables
);
disp('first 10 columns of M:') % to fit on screen
disp(M(:,1:10))
disp('b transposed:') % for pretty
disp(b.')
```

```
first 10 columns of M:
[ 1, -zC1, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 1, -zC2, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 1, -zL1, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 1, -zL2, 0]
[ 0, 0, 0, 0, 0, 0, 0, 1, -zR1]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 1, 0, 0, 0, -1, 0, 0, 0]
[ 0, 0, 0, 0, 0, -1, 0, 0, 1]
[ 0, 0, 0, 1, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 1, 0, 0, 0, 1, 0, 0, 0, 1, 0]
[ 0, 0, -1, 0, 0, 0, 1, 0, 0, 0]
[ -1, 0, 1, 0, 0, 0, 0, 0, 0, 0]

b transposed:
[ 0, 0, 0, 0, 0, 0, 0, 0, -IS, 0, IS, IS, VS, 0, 0]
```

• Furthermore, Matlab has the function `linsolve` for solving for the unknowns.

```
sol_z = linsolve(M,b);
```

This solution `sol_z` includes impedances. We would like to substitute for the actual impedance values, defined as follows in struct form.

```
impedances.zC1 = 1/(C1*s);
impedances.zC2 = 1/(C2*s);
impedances.zL1 = L1*s;
impedances.zL2 = L2*s;
impedances.zR1 = R1;
impedances.zR2 = R2;
impedances.zR3 = R3;
```

Now we can substitute impedances with subs, which gives us a solution in terms of `s`.

```
sol = simplify(...
    subs(...
        sol_z,...
        fieldnames(impedances),...
        struct2cell(impedances)...
    )...
);
[n,d]=numden(sol);
sol_nd = collect([n,d],s);
```

Finally, we can compute the transfer function matrix  $H(s)$  by using the solutions for our outputs and substituting 1 for the input of interest and 0 for the others (this code generalizes to more than two inputs).

```
for input_i = 1:length(in) % each input
    for output_i = 1:length(out_i) % each output
        output_index = out_i(output_i); % idx of ops var
        input_var = in(input_i); % input variable
        other_inputs = setdiff(in,[input_var]); % sneaky
        num = sol_nd(output_index,1); % w/all ip's
        den = sol_nd(output_index,2); % w/all ip's
        num_tf = subs(... % eliminate other inputs
```

```

    num,...
    [input_var,other_inputs],...
    [1,zeros(size(other_inputs))]...
);
den_tf = subs(... % eliminate other inputs
    den,...
    [input_var,other_inputs],...
    [1,zeros(size(other_inputs))]...
);
H(input_i,output_i) = ... % collect s and divide
    collect(num_tf,s)/collect(den_tf,s);
end
end
pretty(H(2,1)) % display H_21

```

$$\frac{(C1 L1 R2 s^2 + C1 R1 R2 s + R2)}{(C1 C2 L1 R2 + C1 C2 L1 R3)}$$

$$s^3 + (C1 L1 + C2 L1 + C1 C2 R1 R2 + C1 C2 R1 R3)$$

$$s^2 + (C1 R1 + C2 R1 + C2 R2 + C2 R3) s + 1)$$

## 13.5 imp.equiv Norton and Thévenin theorems

1 The following remarkable theorem has been proven.<sup>5</sup>

### Theorem 13 imp.1: generalized Thévenin's theorem

Given a linear network of across-variable sources, through-variable sources, and impedances, the behavior at the network's output nodes can be reproduced exactly by a single *across-variable source*  $\mathcal{V}_e$  in series with an impedance  $Z_e$ .

2 The equivalent linear network has two quantities to determine:  $\mathcal{V}_e$  and  $Z_e$ .

*Determining  $Z_e$*

3 The **equivalent impedance**  $Z_e$  of a network is the impedance between the output nodes with all inputs set to zero. Setting an across-variable source to zero means the across-variable on both its terminals are equal, which is equivalent to treating them as the same node. Setting a through-variable source to zero means the through-variable through it is zero, which is equivalent to treating its nodes as disconnected.

*Determining  $\mathcal{V}_e$*

4 The **equivalent across-variable source**  $\mathcal{V}_e$  is the across-variable at the output nodes of the network when they are left open (disconnected from a load). Determining this value typically requires some analysis with the elemental, continuity, and compatibility equations (preferably via impedance methods).

### Norton's theorem

5 Similarly, the following remarkable theorem has been proven.

---

<sup>5</sup>This lecture is intentionally strongly paralleled in our *Electronics* lecture on Norton's and Thévenin's theorems.

**Theorem 13 imp.2: generalized Norton's theorem**

Given a linear network of across-variable sources, through-variable sources, and impedances, the behavior at the network's output nodes can be reproduced exactly by a single *through-variable source*  $\mathcal{F}_e$  in parallel with an impedance  $Z_e$ .

6 The equivalent network has two quantities to determine:  $\mathcal{F}_e$  and  $Z_e$ . The equivalent impedance  $Z_e$  is identical to that of Thévenin's theorem, which leaves the equivalent through-variable source  $\mathcal{F}_e$  to be determined.

*Determining  $\mathcal{F}_e$*

7 The **equivalent through-variable source**  $\mathcal{F}_e$  is the through-variable through the output terminals of the network when they are shorted (collapsed to a single node). Determining this value typically requires some analysis with elemental, continuity, and compatibility equations (preferably via impedance methods).

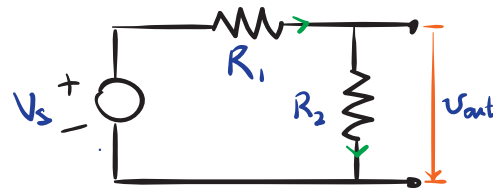
**Converting between Thévenin and Norton equivalents**

8 There is an equivalence between the two equivalent network models that allows one to convert from one to another with ease. The equivalent impedance  $Z_e$  is identical in each and provides the following equation for converting between the two representations:

**Equation 1 converting between Thévenin and Norton equivalents**

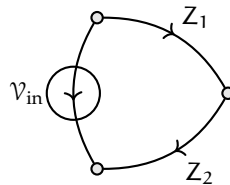
**Example 13.5 imp.equiv-1**

For the circuit shown, find a Thévenin and a Norton equivalent.



re:  
Thévenin  
and  
Norton  
equivalents

## 13.6 imp.divide The divider method



**Figure divide.1:** the two-element cross-variable divider.

**1** In *Electronics*, we developed the useful voltage divider formula for quickly analyzing how voltage divides among series electronic impedances. This can be considered a special case of a more general **across-variable divider** equation for any elements described by an impedance. After developing the across-variable divider, we also introduce the through-variable divider, which divides an input through-variable among parallel elements.

### Across-variable dividers

**2** First, we develop the solution for the two-element across-variable divider shown in [Figure divide.1](#). We choose the across-variable across  $Z_2$  as the output. The analysis follows the impedance method of [Lecture 13.3 imp.tf](#), solving for  $v_2$ .

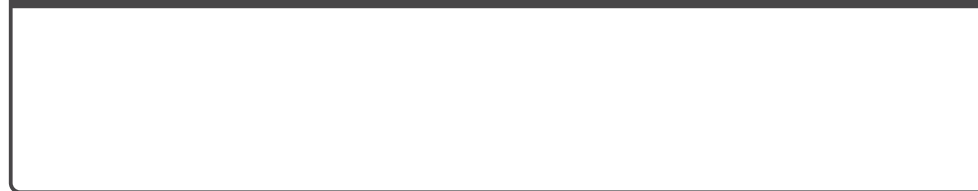
1. Derive four independent equations.
  - a) The normal tree is chosen to consist of  $v_{in}$  and  $Z_2$ .
  - b) The elemental equations are

- c) The continuity equation is
- d) The compatibility equation is

2. Solve for the output  $v_2$ . From the elemental equation for  $Z_2$ ,



- 3 A similar analysis can be conducted for  $n$  impedance elements.

**Equation 1 general across-variable divider****Through-variable dividers**

- 4 By a similar process, we can analyze a network that divides a through-variable into  $n$  *parallel* impedance elements.

**Equation 2 general through-variable divider****Transfer functions using dividers**

- 5 An excellent shortcut to deriving a transfer function is to use the across- and through-variable divider rules instead of solving the system of algebraic equations, as in [Lec. 13.3 imp.tf](#). An algorithm for this process is as follows.



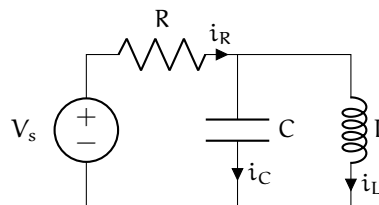
1. Identify the element associated with an output variable  $Y_i$ . Call it the *output element*.
2. Identify the source associated with an input variable  $U_j$ . Set all other sources to zero.
3. Transform the network to be an across- or through-variable divider that includes the “bare” (uncombined) output element’s output variable.<sup>6</sup>
  - a) If necessary, form equivalent impedances of portions of the network, being sure to leave the output element’s output variable alone.
  - b) If necessary, transform the source *à la* Norton or Thévenin.
4. Apply the across- or through-variable divider equation.
5. If necessary, use the elemental equation of the output element to trade output across- and through-variables.
6. If necessary, use the source transformation equation of the input to trade input across- and through-variables.
7. Divide both sides by the input variable.

**6** It turns out that, despite its many “if necessary” clauses, very often this “shortcut” is easier than the method of [Lecture 13.3 imp.tf](#) for low-order systems if only a few transfer functions are of interest.

### Example 13.6 imp.divide-1

Given the circuit shown with voltage source  $V_s$  and output  $v_L$ ,

- a. what is the transfer function  $\frac{V_L}{V_s}$ ?
- b. Without transforming the source, find the transfer function  $\frac{I_L}{V_s}$ .
- c. Transforming the source, find  $\frac{I_L}{V_s}$ .



re: a  
circuit  
transfer  
function  
using  
a  
divider

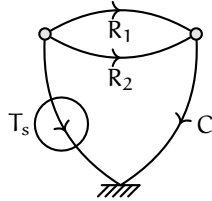
<sup>6</sup>In other words, if the across-variable of the output element is the output, do not combine it in series; if the through-variable is the output, do not combine it in parallel.



## 13.7 imp.exe Exercises for Chapter 13 imp

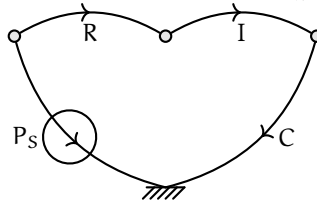
### Exercise 13.1 file

Use the linear graph below of a thermal system to (a) derive the transfer function  $T_{R_2}(s)/T_s(s)$ , where  $T_s$  is the input temperature and  $T_{R_2}$  is the temperature across the thermal resistor  $R_2$ . Use *impedance methods*. And (b) derive the input impedance the input  $T_s$  drives.



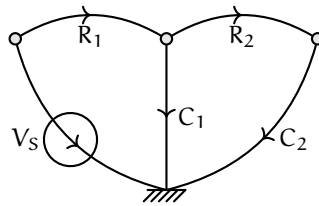
### Exercise 13.2 granite

Use the linear graph below of a fluid system to (a) derive the transfer function  $P_C(s)/P_S(s)$ , where  $P_S$  is the input pressure and  $P_C$  is the pressure across the fluid capacitance  $C$ . Use *impedance methods and a divider rule is highly recommended*. (Simplify the transfer function.) And (b) derive the input impedance the input  $P_S$  drives. (Don't simplify the expression.)



### Exercise 13.3 granted

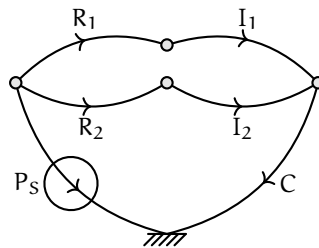
Use the linear graph below of an electronic system to derive the transfer function  $I_{R_1}(s)/V_S(s)$ , where  $V_S$  is the input voltage and  $I_{R_1}$  is the current through the resistor  $R_1$ . (Simplify the transfer function.) Use an impedance method. *Hint: a divider method is recommended; without it, use of a computer is recommended.*



**Exercise 13.4 concrete**

Use the linear graph of a fluid system in Fig. exe.1 to derive the transfer function  $Q_C(s)/P_S(s)$ , where  $P_S$  is the input pressure and  $Q_C$  is the flowrate through the fluid capacitance  $C$ . Use impedance methods; a divider rule is recommended but not required. Identify all impedances but do not substitute them into the transfer function.

\_\_\_\_\_/ 25 p.

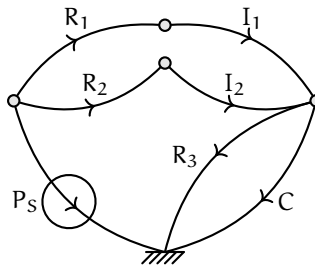


**Figure exe.1:** a fluid system linear graph.

**Exercise 13.5 rhine**

Use the linear graph of a fluid system in Fig. exe.2 to derive the transfer functions  $Q_C(s)/P_S(s)$  and  $Q_{R_3}(s)/P_S(s)$ , where  $P_S$  is the input pressure,  $Q_{R_3}$  is the flowrate through a valve with resistance  $R_3$ , and  $Q_C$  is the flowrate through a tank with fluid capacitance  $C$ . Use impedance methods; a divider rule is not required. Identify all impedances and substitute them into the transfer functions, but you are not required to simplify these expressions.

\_\_\_\_\_/ 30 p.



**Figure exe.2:** a fluid system linear graph.

### Exercise 13.6 tableau

Consider an accelerometer that has transfer function

$$G(s) \equiv \frac{V_i(s)}{A(s)} = \frac{K_G \omega_{n_G}^2}{s^2 + 2\zeta_G \omega_{n_G} s + \omega_{n_G}^2}, \quad (1)$$

where

- $A$  is the input acceleration in  $\text{m/s}^2$ ,
- $V_i$  is the output voltage in  $\text{V}$ ,
- $K_G = 0.1 \text{ V}/(\text{m/s}^2)$  is the gain,
- $\omega_{n_G} = 3000 \text{ rad/s}$  is the natural frequency, and
- $\zeta_G = 0.2$  is the damping ratio.

Perform a frequency domain analysis as follows.

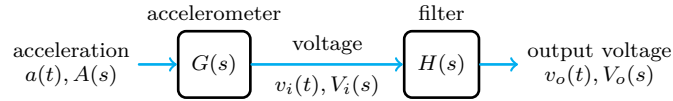
- a. Generate a Bode plot of  $G(s)$ .
- b. At DC ( $\omega = 0 \text{ rad/s}$ ), compute the *magnitude* and *phase* of the frequency response function of the accelerometer.

Suppose there is a sinusoidal systematic noise signal at the input, with amplitude  $a_{\text{noise}} = 1 \text{ m/s}^2$  and frequency  $\omega_{\text{noise}} = 2900 \text{ rad/s}$ .<sup>7</sup>

- c. Assuming there is only noise input, at the noise frequency  $\omega_{\text{noise}}$ , compute the *amplitude* and *phase* of the voltage  $V_i$  at the output of the accelerometer. Why is the amplitude higher than it would have been at DC (use your Bode plot from **Item a.** to justify your answer).

<sup>7</sup>Assume the input phase is zero.

To mitigate the systematic noise, we add a filter with transfer function  $H(s)$  to the output of the accelerometer, as shown in Fig. exe.3.

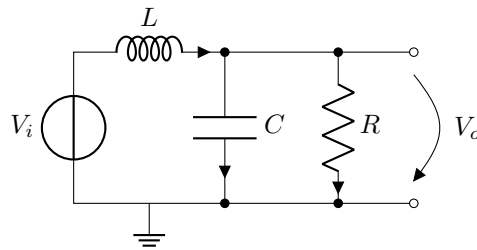


**Figure exe.3:** Accelerometer and filter block diagram.

By definition,

$$H(s) \equiv \frac{V_o(s)}{V_i(s)}. \quad (2)$$

Assume the filter and accelerometer *do not* dynamically load each other. The filter circuit diagram is shown in Fig. exe.4.



**Figure exe.4:** Filter circuit.

- d. Draw a linear graph model of the filter circuit.
- e. Use *impedance methods* to derive the transfer function  $H(s)$  in terms of the circuit element parameters  $R$ ,  $L$ , and  $C$ .
- f. Find the filter's natural frequency  $\omega_{nH}$  and damping ratio  $\zeta_H$ .<sup>8</sup>
- g. Let  $C = 0.001$  F. Design the filter by choosing  $R$  and  $L$  such that

$$\zeta_H = 1 \quad \text{and} \quad \omega_{nH} = 1000 \text{ rad/s}. \quad (3)$$

- h. Find the transfer function

$$\frac{V_o(s)}{A(s)} \quad (4)$$

with all parameters substituted. Simplify.

<sup>8</sup>Be cautious to make the denominator have the proper standard form  $s^2 + 2\zeta_H\omega_{nH}s + \omega_{nH}^2$ .

- i. Generate a Bode plot for  $V_o(s)/A(s)$ .
- j. Using the Bode plot of **Item i.**, explain why we should expect the output from the systematic noise at  $\omega_{\text{noise}}$  to be improved.
- k. From the transfer function  $V_o(s)/A(s)$ , at the noise frequency  $\omega_{\text{noise}}$ , compute the *amplitude* and *phase* of the output voltage  $V_o$ .
- l. Compare the result from **Item k.** to the unfiltered voltage in **Item c.** by finding the ratio of the filtered amplitude over then unfiltered amplitude.
- m. How could you augment the filter design to further reduce the systematic noise?

### Exercise 13.7 gypsum

Respond to the following questions and imperatives with a sentence or two, equation, and/or a sketch.

- a. Comment on the *stability* and *transient response characteristics* of a system with eigenvalues

$$-2, -5, -8 + j3, -8 - j3.$$

- b. Consider an LTI system that, given input  $u_1$ , outputs  $y_1$ , and given input  $u_2$ , outputs  $y_2$ . If the input is  $u_3 = 5u_1 - 6u_2$ , what is the output  $y_3$ ?
- c. Consider a second-order system with natural frequency  $\omega_n = 2$  rad/s and damping ratio  $\zeta = 0.5$ . What is the free response for initial condition  $y(0) = 1$ ?
- d. Two thermal elements with impedances  $Z_1$  and  $Z_2$  have a temperature source  $T_S$  applied across them *in series*. What is the transfer function from  $T_S$  to the heat  $Q_2$  through  $Z_2$ ?
- e. Draw a linear graph of a pump (pressure source) flowing water through a long pipe into the bottom of a tank, which has a valve at its bottom from which the water flows.

## **Part VI**

# **Nonlinear system analysis**



- 1 Thus far, we've mostly considered *linear* system models. Many of the analytic tools we've developed—ODE solution techniques, superposition, eigendecomposition, stability analysis, impedance modeling, transfer functions, frequency response functions—do not apply to nonlinear systems. In fact, analytic solutions are unknown for most nonlinear system ODEs. And even basic questions are relatively hard to answer; for instance: is the system stable?
- 2 In this and the following chapters, we consider a few analytic and numerical techniques for dealing with nonlinear systems.
- 3 A state-space model has the general form

$$\frac{dx}{dt} = f(\mathbf{x}, \mathbf{u}, t) \tag{1a}$$

$$\mathbf{y} = \underline{\hspace{3cm}} \tag{1b}$$

where  $f$  and  $g$  are vector-valued functions that depend on the system.

**Nonlinear state-space models** are those for which  $f$  is a \_\_\_\_\_ functional of either  $\mathbf{x}$  or  $\mathbf{u}$ . For instance, a state variable  $x_1$  might appear as  $x_1^2$  or two state variables might combine as  $x_1x_2$  or an input  $u_1$  might enter the equations as  $\log u_1$ .

### Autonomous and nonautonomous systems

- 4 An **autonomous system** is one for which  $f(\mathbf{x})$ , with neither time nor input appearing explicitly. A **nonautonomous system** is one for which either  $t$  or  $\mathbf{u}$  *do* appear explicitly in  $f$ . It turns out that we can always write

nonautonomous systems as autonomous by substituting in  $\mathbf{u}(t)$  and introducing an extra \_\_\_\_\_ for  $t^1$ .

5 Therefore, without loss of generality, we will focus on ways of analyzing autonomous systems.

## Equilibrium

6 An **equilibrium state** (also called a \_\_\_\_\_)  $\bar{\mathbf{x}}$  is one for which  $dx/dt = 0$ . In most cases, this occurs only when the input  $\mathbf{u}$  is a constant  $\bar{\mathbf{u}}$  and, for time-varying systems, at a given time  $\bar{t}$ . For autonomous systems, equilibrium occurs when the following holds:

$$\text{_____} \tag{2}$$

This is a system of nonlinear algebraic equations, which can be challenging to solve for  $\bar{\mathbf{x}}$ . However, frequently, several solutions—that is, equilibrium states—do exist.

---

<sup>1</sup> Strogatz and Dichter, 2016.

## 14.1 nlin.lin Linearization

**1** A common method for dealing with a nonlinear system is to **linearize** it: transform it such that its state equation is linear. A linearized model is typically only valid in some neighborhood of state-space. This neighborhood is selected by choosing an **operating point**  $\mathbf{x}_o$  used in the linearization process. We use two considerations when choosing an operating point:

1. that implied by the name—it should be in a region of state-space in which the state will stay throughout the system's operation—and
2. the validity of the model near the operating point.

Due to the fact that nonlinear systems tend to be more-linear near equilibria, the second consideration frequently suggests we choose one as an operating point:  $\mathbf{x}_o = \bar{\mathbf{x}}$ .

### Taylor series expansion

**2** A Taylor series expansion of Eq. 1a about an operating point  $\mathbf{x}_o, \mathbf{u}_o$  (for a nonautonomous system) yields polynomial terms that are linear, quadratic, etc. in  $\mathbf{x}$  and  $\mathbf{u}$ . If we keep only the linear terms and define new state and input variables

$$\mathbf{x}^* = \mathbf{x} - \mathbf{x}_o \quad \text{and} \quad \mathbf{u}^* = \mathbf{u} - \mathbf{u}_o \quad (1a)$$

we get a linear state equation

$$\frac{d\mathbf{x}^*}{dt} = \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u}^* \quad (1b)$$

where the matrix components are given by

$$A_{ij} = \left. \frac{\partial f_i}{\partial x_j} \right|_{\mathbf{x}_o, \mathbf{u}_o} \quad \text{and} \quad B_{ij} = \left. \frac{\partial f_i}{\partial u_j} \right|_{\mathbf{x}_o, \mathbf{u}_o}. \quad (1c)$$

These first-derivative matrices are generally called **Jacobian** matrices.

**3** This result also applies to autonomous equations if we drop the  $\mathbf{B}\mathbf{u}^*$  term.

**Example 14.1 nlin.lin-1****re:  
hardening  
spring**

Consider a vehicle suspension system that is overloaded such that its springs are exhibiting *hardening* behavior such that a lumped-parameter constitutive equation for the springs (collectively) is

$$f_k = kx_k + \alpha x_k^3 \quad (2)$$

where  $f_k$  is the force,  $x_k$  the displacement, and  $k, \alpha > 0$  constant parameters of the spring.

- Develop a (nonlinear) spring-mass-damper linear graph model for the vehicle suspension with input position source  $X_s$ .
- Derive a nonlinear state-space model from the linear graph model using the state vector

$$\mathbf{x} = [x_m \quad v_m]^T. \quad (3)$$

- Linearize the system about the operating point

$$\mathbf{x}_o = [1 \quad 0]^T \quad \text{and} \quad \mathbf{u}_o = [0] \quad (4)$$

by computing the A, B, and E matrices of the linearized system.<sup>a</sup>

<sup>a</sup>The E matrix is the Jacobian with respect to the time-derivative of the input:  $\dot{\mathbf{u}}$ , which arises occasionally.

## 14.2 nlin.char Nonlinear system characteristics

1 Characterizing nonlinear systems can be challenging without the tools developed for \_\_\_\_\_ system characterization. However, there are ways of characterizing nonlinear systems, and we'll here explore a few.

### Those in-common with linear systems

2 As with linear systems, the **system order** is either the number of state-variables required to describe the system or, equivalently, the highest-order \_\_\_\_\_ in a single scalar differential equation describing the system.

3 Similarly, nonlinear systems can have state variables that depend on \_\_\_\_\_ alone or those that also depend on \_\_\_\_\_ (or some other independent variable). The former lead to ordinary differential equations (ODEs) and the latter to partial differential equations (PDEs).

4 Equilibrium was already considered in [Chapter 14 nlin](#).

### Stability

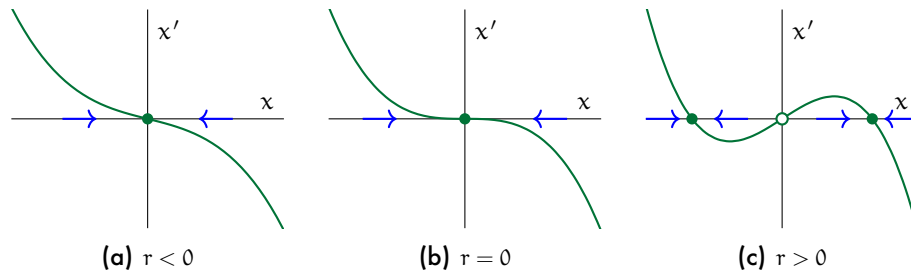
5 In terms of system performance, perhaps no other criterion is as important as \_\_\_\_\_.

#### Definition 14 nlin.1: Stability

If  $x$  is perturbed from an equilibrium state  $\bar{x}$ , the response  $x(t)$  can:

1. asymptotically return to  $\bar{x}$  (asymptotically \_\_\_\_\_),
2. diverge from  $\bar{x}$  (\_\_\_\_\_), or
3. remain perturbed or oscillate about  $\bar{x}$  with a constant amplitude (\_\_\_\_\_ stable).

Notice that this definition is actually local: stability in the neighborhood of one equilibrium may not be the same as in the neighborhood of another.



**Figure char.1:** plots of  $x'$  versus  $x$  for Eq. 1.

6 Other than nonlinear systems' lack of linear systems' eigenvalues, poles, and roots of the characteristic equation from which to compute it, the primary difference between the stability of linear and nonlinear systems is that nonlinear system stability is often difficult to establish. Using a linear system's eigenvalues, it is straightforward to establish stable, unstable, and marginally stable subspaces of state-space (via transforming to an eigenvector basis). For nonlinear systems, no such method exists. However, we are not without tools to explore nonlinear system stability. One mathematical tool to consider is Lyapunov's method, which is beyond the scope of this course, but has good treatments in<sup>2</sup> and<sup>3</sup>.

**Qualities of equilibria**

7 Equilibria (i.e. stationary points) come in a variety of qualities. It is instructive to consider the first-order differential equation in state variable  $x$  with real constant  $r$ :

$$x' = rx - x^3. \tag{1}$$

If we plot  $x'$  versus  $x$  for different values of  $r$ , we obtain the plots of Fig. char.1.

8 By definition, equilibria occur when  $x' = 0$ , so the  $x$ -axis crossings of Fig. char.1 are equilibria. The blue arrows on the  $x$ -axis show the direction of state change  $x'$ , quantified by the plots. For both

<sup>2</sup> Brogan, 1991, Ch. 10.  
<sup>3</sup> Choukchou-Braham and others, 2013, App. A.

(a) and (b), only one equilibrium exists:  $x = 0$ . Note that the blue arrows in both plots point *toward* the equilibrium. In such cases—that is, when a \_\_\_\_\_ exists around an equilibrium for which state changes point toward the equilibrium—the equilibrium is called an \_\_\_\_\_ or \_\_\_\_\_. Note that attractors are \_\_\_\_\_.

9 Now consider (c) of Fig. char.1. When  $r > 0$ , three equilibria emerge. This change of the number of equilibria with the changing of a parameter is called a \_\_\_\_\_. A plot of bifurcations versus the parameter is called a **bifurcation diagram**. The  $x = 0$  equilibrium now has arrows that point \_\_\_\_\_ from it. Such an equilibrium is called a \_\_\_\_\_ or \_\_\_\_\_ and is \_\_\_\_\_. The other two equilibria here are (stable) attractors. Consider a very small initial condition  $x(0) = \epsilon$ . If  $\epsilon > 0$ , the repeller pushes away  $x$  and the positive attractor pulls  $x$  to itself. Conversely, if  $\epsilon < 0$ , the repeller again pushes away  $x$  and the negative attractor pulls  $x$  to itself.

10 Another type of equilibrium is called the \_\_\_\_\_: one which acts as an attractor along some lines and as a repeller along others. We will see this type in the following example.

**Example 14.2 nlin.char-1**

Consider the dynamical equation

$$x' = x^2 + r \tag{2}$$

with  $r$  a real constant. Sketch  $x'$  vs  $x$  for negative, zero, and positive  $r$ . Identify and classify each of the equilibria.

re:  
**Saddle  
bifurcation**

## 14.3 nlin.exe Exercises for Chapter 14 nlin

### Exercise 14.1 sigmund

Consider a nonlinear capacitor with constitutive equation relating charge  $q_C$  and voltage  $v_C$ :

$$q_C = kv_C^{3/2} \quad (1)$$

with  $k$  a positive constant.

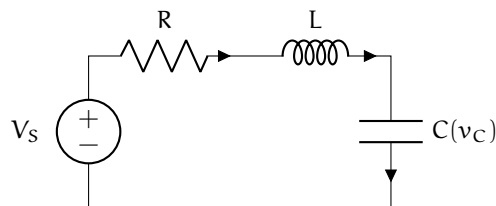
- Derive an elemental equation relating  $dv_C/dt$  and  $i_C$  for the nonlinear capacitor.
- From the elemental equation, what is the voltage-dependent capacitance  $C(v_C)$ ?
- Consider the RLC-circuit of Fig. exe.1, which includes the nonlinear capacitor. Derive a nonlinear state-space equation with state vector

$$\mathbf{x} = [v_C \quad i_L]^T. \quad (2)$$

- For a constant input  $V_S(t) = 5 \text{ V}$ , derive the equilibrium state.
- Linearize the state-space equation about the operating point

$$\mathbf{x}_o, \mathbf{u}_o = [5 \text{ V} \quad 0 \text{ A}]^T, [5 \text{ V}]. \quad (3)$$

Define the state equation matrices  $A$  and  $B$ , the linearized state and input vectors  $\mathbf{x}^*$  and  $\mathbf{u}^*$ , and the linearized state equation.



**Figure exe.1:** circuit for Exercise 14.1 nlin..

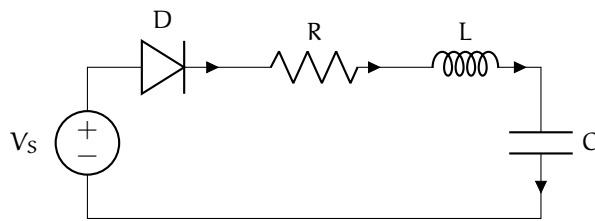


**Exercise 14.2 franz**

A nonlinear diode model gives a diode's elemental equation to be

$$i_D = I_s(\exp(v_D/V_{TH}) - 1).$$

We let the saturation current be  $I_s = 10^{-12}$  A and the thermal voltage be  $V_{TH} = 0.025$  V. Considering this nonlinear diode model for the circuit of Fig. exe.2.



**Figure exe.2:** circuit for Exercise 14.2 nlin..

- a. Derive a nonlinear state-space equation with state vector

$$\mathbf{x} = [v_C \quad i_L]^T. \quad (4)$$

*Hint: include the diode in your normal tree.*

- b. For a constant input  $V_s(t) = 0$  V, derive the equilibrium state.  
 c. Linearize the state-space equation about the operating point

$$\mathbf{x}_o, \mathbf{u}_o = [0\text{ V} \quad 0\text{ A}]^T, [0\text{ V}]. \quad (5)$$

*Hint:  $d \ln(x)/dx = 1/x$ . Define the state equation matrices A and B, the linearized state and input vectors  $\mathbf{x}^*$  and  $\mathbf{u}^*$ , and the linearized state equation.*

**15 phase**

---

**16 sim**

---

## 16.1 `sim.python` Nonlinear Systems in Python

Most of the Python Control Systems package tools we've used will not work for nonlinear systems. For instance, nonlinear systems cannot be defined with `control.tf()`, `control.ss()`, and `control.zpk()`. Similarly, the simulation functions `control.forced_response()`, `control.initial_response()`, and `control.step_response()` do not work for nonlinear systems.

There are two common ways of defining and simulating nonlinear systems in Python. The first uses the SciPy package's `integrate` module's functions such as `solve_ivp`. The second uses the Control Systems package, which has nonlinear state-space model representations. For simulating nonlinear systems, the Control Systems package actually calls the SciPy package's `integrate` module's functions. Because we have already been using the Control Systems package for linear system models, we will use its nonlinear facilities, as well. However, it should be mentioned that the package's documentation for nonlinear systems is a bit sparse.

### Defining a Nonlinear System

We can define a nonlinear system in the Control Systems package by calling the `control.nlsys()` function. Here are its most important arguments, for us:

- `updfcn` (callable): The state update function that encodes the right-hand side of the state equation (i.e.,  $f(\mathbf{x}, \mathbf{u}, t)$ ). It should have the form `updfcn(t, x, u, params) -> array`, where `t` is the current value of time, `x` is a 1D NumPy array representing the states, `u` is a 1D array representing the inputs, and `params` is a `dict` of parameter values. The function should return an array of state derivatives (i.e.,  $\mathbf{x}'$ ).
- `outfcn` (callable, optional): The output function that encodes the right-hand side of the output equation (i.e.,  $g(\mathbf{x}, \mathbf{u}, t)$ ). It should have

the form `outfcn(t, x, u, params) -> array`, where `t`, `x`, `u`, and `params` are as they were for `updfcn`. If this argument is not provided, the output is taken to be the states (i.e.,  $\mathbf{y} = \mathbf{x}$ ).

- `inputs` (`int`, `list` of `str` or `None`, optional): System inputs description. The number of inputs is given by an `int`. A name for each can be given as a `list` of `strs`. If it is not provided or if `None` is passed, the function will attempt to discern the inputs.
- `outputs` (`int`, `list` of `str` or `None`, optional): System outputs description, with the same options as inputs.
- `states` (`int`, `list` of `str` or `None`, optional): System states description, with the same options as inputs.
- `params` (`dict`, optional): Numerical values of parameters for evaluation in functions.

Consider the Van der Pol oscillator nonlinear state-space model

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x}) \\ &= \begin{bmatrix} x_2 \\ (1 - x_1^2)x_2 - x_1 \end{bmatrix}.\end{aligned}\tag{1}$$

Note that this is an autonomous system (i.e., there are no inputs). This state equation has applications in electrical and biological modeling. We can encode its dynamics in the following Python update function:

```
def van_der_pol_update(t, x, u, params):
    """Returns the rhs of the Van der Pol state equation"""
    dxdt = np.array([x[1], (1 - x[0]**2) * x[1] - x[0]])
    return dxdt
```

Now we can create a nonlinear system model with `control.nlsys()` as follows:

```
sys = control.nlsys(van_der_pol_update, inputs=0, states=2, outputs=2)
```

This creates a `NonlinearIOSystem` object.

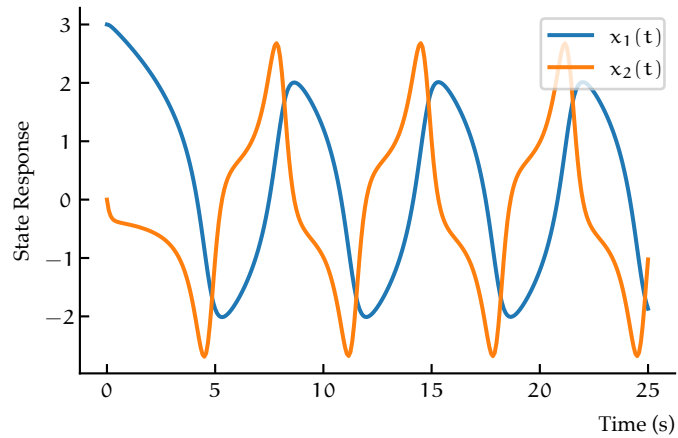
**Simulating a Nonlinear System** A nonlinear system in the form of a `NonlinearIOSystem` object can be simulated (i.e., numerically solved) with the `control.input_output_response()` function. This function is very similar to `control.forced_response()`, so we will immediately apply it to our Van der Pol oscillator model as follows:

```
T = np.linspace(0, 25, 301) # Simulation time array
y = control.input_output_response(
    sys, T=T, X0=[3, 0], squeeze=True
).outputs
```

This returns a `TimeResponseData` object, just as does `control.forced_response()`, so we have selected the `outputs` data attribute.

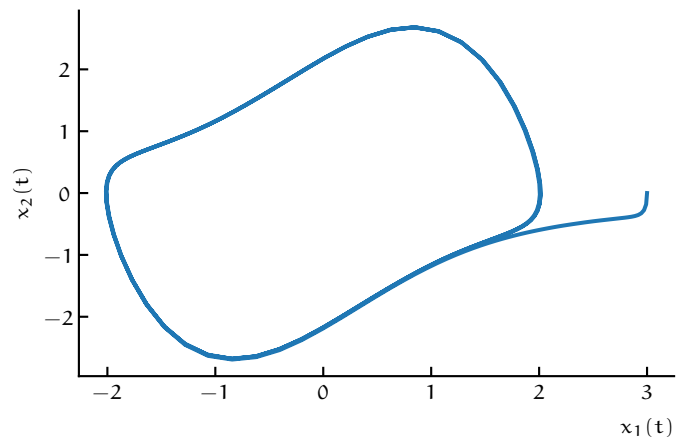
**Plotting the Step Response** We can plot the response through time as follows:

```
fig, ax = plt.subplots()
ax.plot(T, y[0,:], label="$x_1(t)$")
ax.plot(T, y[1,:], label="$x_2(t)$")
ax.set_xlabel("Time (s)")
ax.set_ylabel("State Response")
ax.legend(loc='upper right')
plt.show()
```



**Figure python.1:** A state free response through time.

```
fig, ax = plt.subplots()
ax.plot(y[0,:], y[1,:])
ax.set_xlabel("$x_1(t)$")
ax.set_ylabel("$x_2(t)$")
plt.show()
```



**Figure python.2:** A phase plot of the free response.

## 16.2 `sim.matlab` Nonlinear systems in Matlab

Many of the Matlab tools we've used will not work for nonlinear systems; for instance, system-definition with `tf`, `ss`, and `zpk` and simulation with `lsim`, `step`, `initial`—none will work with nonlinear systems.

### Defining a nonlinear system

We can define a nonlinear system in Matlab by defining its state-space model in a function file. Consider the nonlinear state-space model<sup>1</sup>

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) \\ &= \begin{bmatrix} x_2 \\ (1 - x_1^2)x_2 - x_1 \end{bmatrix}. \end{aligned} \tag{1}$$

A function file describing it is as follows.

```
type van_der_pol.m
```

```
function dxdt = van_der_pol(t,x)
    dxdt = [ ...
            x(2); ...
            (1-x(1)^2)*x(2) - x(1) ...
            ];
```

Note that  $\mathbf{x}$  is representing the (two) state vector  $\mathbf{x}$ , which, along with time  $t$  ( $t$ ), are passed as arguments to `van_der_pol`. The variable `dxdt` serves as the output (return) of the function. Effectively, `van_der_pol` is simply  $\mathbf{f}(\mathbf{x})$ , the right-hand side of the state equation.

### Simulating a nonlinear system

The nonlinear state equation is a system of ODEs. Matlab has several numerical ODE solvers that perform well for nonlinear systems. When

<sup>1</sup>This is a van der Pol equation.



choosing a solver, the foremost considerations are **ODE stiffness** and **required accuracy**. Stiffness occurs when solutions evolve on drastically different time-scales. For a more-thorough guide for selecting an ODE solver, see

[mathworks.com/help/matlab/math/choose-an-ode-solver.html](https://mathworks.com/help/matlab/math/choose-an-ode-solver.html).

For most ODEs, the `ode45` Runge-Kutta solver is the best choice, so try it first. Its syntax is paradigmatic of all Matlab solvers.

```
[t,y] = ode45( ...  
    odefun, ... % ODE function handle, e.g. van_der_pol  
    time, ...   % time array or span  
    x0 ...      % initial state  
)
```

Details here include

1. the ODE function given must have exactly two arguments: `t` and `x`;
2. the time array or span doesn't impact solver steps; and
3. the initial conditions must be specified in a vector size matching the state vector `x`.

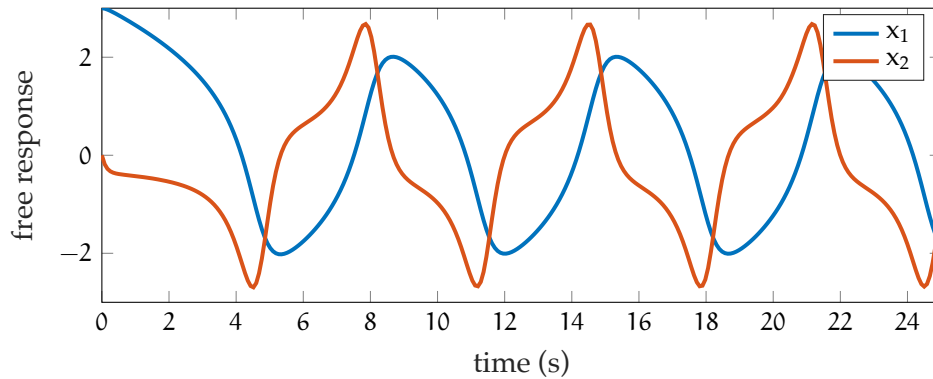
Let's apply this to our example from above. We begin by specifying the simulation parameters.

```
x0 = [3;0];  
t_a = linspace(0,25,300);
```

And now we simulate.

```
[~,x] = ode45(@van_der_pol,t_a,x0);
```

Note that since we specified a full time array `t_a`, and not simply a range, the time (first) output is superfluous. We can avoid assigning it a variable by inserting `~` appropriately.



**Figure matlab.1:** free response plotted through time.

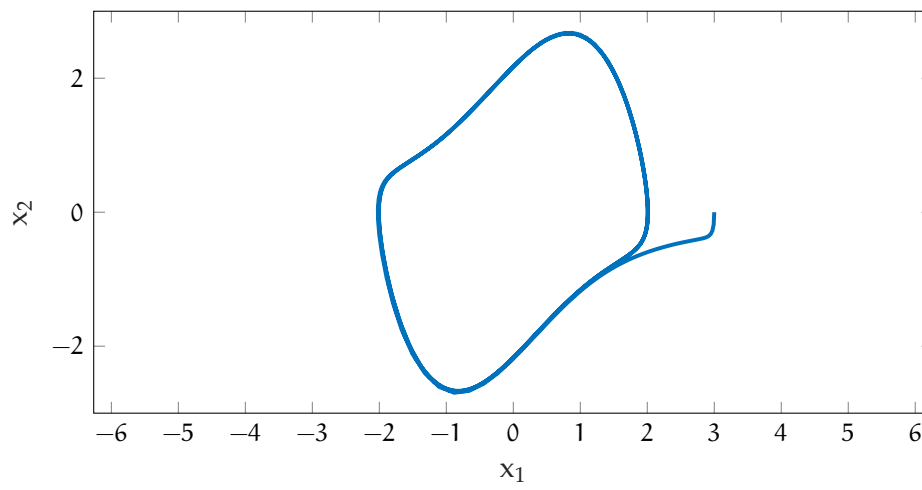
### Plotting the response

In time, the response is shown in [Fig. matlab.1](#). Note the weirdness—this is certainly no decaying exponential!

```
figure
plot( ...
    t_a,x.', ...
    'linewidth',1.5 ...
)
xlabel('time (s)')
ylabel('free response')
legend('x_1','x_2')
```

It seems the response is settling into a non-sinusoidal periodic function. This is especially obvious if we consider the phase portrait of [Fig. matlab.2](#).

```
figure
plot( ...
    x(:,1),x(:,2), ...
    'linewidth',2 ...
)
xlabel('x_1')
ylabel('x_2')
```



**Figure matlab.2:** free response plotted in phase space.

## 16.3 sim.fluid Nonlinear fluid system example

- 1 This example gets one started on the design problem [Exercise 16.3 sim.](#).
- 2 Consider a fluid system with an input volumetric flowrate  $Q_s$  into a capacitance  $C$  that is drained by only a single pipe of *nonlinear* resistance  $R$  and  $L$ , as shown in the linear graph of [Figure fluid.1](#). The nonlinearity of  $R$  is a good way to model an overflow. In this lecture, we will derive a **nonlinear state-space model** for the system—specifically, a state equation—and solve it, numerically using Matlab.

### Normal tree, order, and variables

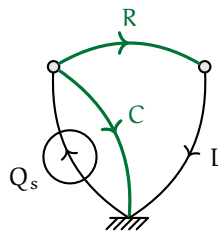
- 3 [Fig. fluid.1](#) already shows the normal tree. There are two independent energy storage elements, making it a second-order ( $n = 2$ ) system. We define the state vector to be

$$\mathbf{x} = [P_C \quad Q_L]^T. \quad (1)$$

The input vector is defined as  $\mathbf{u} = [Q_s]$ .

### Elemental, continuity, and compatibility equations

- 4 Before turning to our familiar elemental equations, we'll consider the nonlinear resistor.



**Figure fluid.1:** a linear graph and normal tree (green) for a nonlinear fluid system.

*Nonlinear elemental equation*

5 Suppose we are trying to model an overflow with the pipe R–L to ground. An overflow would have no flow until the fluid capacitor fills to a certain height, then it would transition to flowing quite rapidly. This process seems to be inherently nonlinear because we cannot write an element that depends linearly on the height of the fluid in the capacitor (even if height was one of our state variables, which it is not).

6 The volume in the tank can be found by integrating in flow ( $Q_s$ ) minus out flow ( $Q_R$ ), but this is not accessible within a simulation, since it must be integrated, so it's not an ideal variable for our model. However, the pressure  $P_C$ —a state variable—is proportional to the fluid height in the capacitor, which we'll call  $h$ :

$$P_C = \rho gh, \quad (2)$$

where  $\rho$  is the density of the fluid and  $g$  is the gravitational acceleration. Since the height of the capacitor is presumably known, we can use  $P_C$  to be our fluid height metric.

7 When the height  $h$  reaches a certain level, probably near the capacitor's max, which we'll denote  $h_m$ , we want our overflow pipe R–L to start flowing. Since  $P_C$  is our height metric, we want to define a resistance as a function of it,  $R(P_C)$ .

8 Now we must determine the form of  $R(P_C)$ . Clearly, when  $h \sim P_C$  is small, we want as little as possible flow through R–L, so  $R(P_C)$  should be large. If  $R$  was infinitely large, divisions by zero would likely arise in a simulation, so we choose to set our low-pressure  $R$  to some finite value:

$$R(P_C)|_{P_C \rightarrow 0} = R_0. \quad (3)$$

Conversely, when  $h \sim P_C$  is large (near max), we want maximum flow through R–L, so  $R(P_C)$  should be some finite value, say, that of the pipe:

$$R(P_C)|_{P_C \rightarrow \infty} = R_\infty. \quad (4)$$

Clearly, this model requires  $R_\infty \ll R_0$ .

9 The transition from  $R_0$  to  $R_\infty$  should be smooth in order to minimize numerical solver difficulties. Furthermore, a smooth transition is consistent with, say, a float opening a valve at the bottom of the capacitor,<sup>2</sup> since the valve would transition continuously from closed to open. Many functions could be used to model this transition, especially if piecewise functions are considered. However, the tanh function has the merit of enabling us to easily define a single non-piecewise function for the entire domain. Let  $\bar{P}_C$  be the transition pressure and  $\Delta P_C$  be the transition width. A convenient nonlinear resistor, then, is

$$R(P_C) = R_\infty + \frac{R_0 - R_\infty}{2} \left( 1 - \tanh \frac{5(P_C - \bar{P}_C)}{\Delta P_C} \right). \quad (5)$$

Note that this function only approximately satisfies  $R(P_C)|_{P_C \rightarrow 0} = R_0$ , but the small deviation from this constraint is worth it for the convenience it provides. Another noteworthy aspect of Equation 5 is the factor of 5, which arises from the tanh function's natural transition width, which we alter via  $\Delta P_C$ .

*Other elemental equations and the continuity and compatibility equations*

10 The other elemental equations have been previously encountered and are listed in the table, below. Furthermore, continuity and compatibility equations can be found in the usual way—by drawing contours and temporarily creating loops by including links in the normal tree. We proceed by drawing a table of all elements and writing an elemental equation for each element, a continuity equation for each branch of the normal tree, and a compatibility equation for each link.

---

<sup>2</sup>Note that this model might be said to assume the overflow pipe is attached to the bottom of the capacitor since the pressure driving fluid through this pipe is supposed to be  $P_C$ . However, no matter the overflow valve's inlet height, if its outlet is at the height of the bottom of the capacitor, this model is still valid.

el.	elemental eq.	el.	cont/comp. eq.
C	$\frac{dP_C}{dt} = \frac{1}{C}Q_C$	C	$Q_C = Q_s - Q_L$
L	$\frac{dQ_L}{dt} = \frac{1}{L}P_L$	L	$P_L = P_C - P_R$
R	$P_R = Q_R R(P_C)$	R	$Q_R = Q_L$

### State equation

**11** The system of equations composed of the elemental, continuity, and compatibility equations can be reduced to the state equation. This equation *nonlinear*, so it cannot be written in the linear form with A and B matrices. However, it can still be written as a system of first-order ordinary differential equations, as follows:

$$\begin{aligned} \frac{dx}{dt} &= f(x, \mathbf{u}) \\ &= \begin{bmatrix} (Q_s - Q_L)/C \\ (P_C - Q_L R(P_C))/L \end{bmatrix}. \end{aligned} \quad (6)$$

Although it appears simple, this nonlinear differential equation likely has no known analytic solution. Two other options are available:

1. linearize the model about an operating point and solve the linearized equation or
2. numerically solve the nonlinear equation.

Both methods are widely useful, but let's assume we require the model to be accurate over a wide range of capacitor fullness. Therefore, we choose to investigate via numerical solution.

## Simulation

Broadly, the numerical investigation will be conducted via Matlab's ode23t solver.<sup>3</sup> Of course, as with any numerical solution, specific values of the parameters must be selected. We begin with declaring the fluid to be water, endowing it with a density, and specify the gravitational acceleration  $g$ . Furthermore, an "anonymous" function  $P\_fun$  is defined, accepting the height  $h$  of the fluid in the capacitor and returning the corresponding pressure. Other parameters specified include the fluid capacitance  $C$  and the overflow pipe inertance  $L$ .

```
global C L % global to be used in state equation
C = 1e3; % ... fluid capacitance
L = 1e-3; % ... fluid inertance
rho = 997; % kg/m3 ... density of water
g = 9.81; % m/s2 ... gravitational constant
P_fun = @(h) rho*g*h; % pressure as a function of height
```

Next, we define the maximum height  $h\_max$  of fluid in the capacitor, the transition height  $h\_t$ , and the distance  $dh$  over which the resistor will transition from high to low impedance.

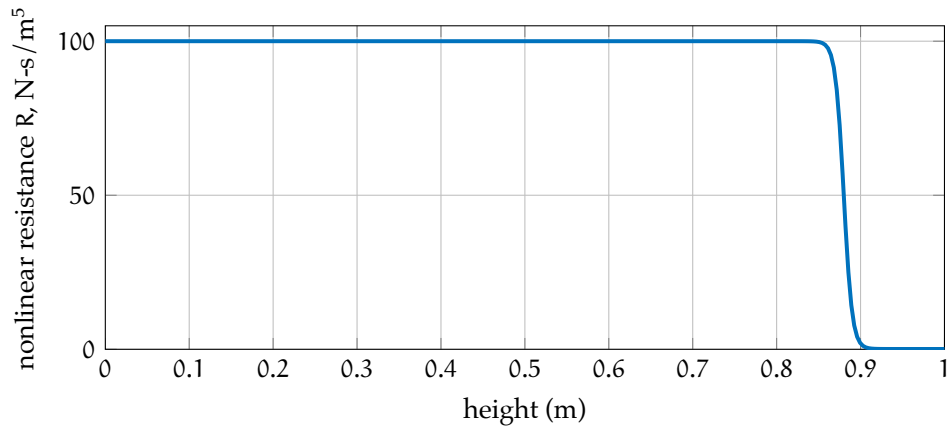
```
h_max = 1; % m ... maximum height of fluid
h_t = .88; % m ... transition height
dh = .05; % m ... height difference for transition
```

Corresponding pressures, which we prefer for computation, can be computed with  $P\_fun$ .

```
P_t = P_fun(h_t); % N/m2 ... transition pressure
dP = P_fun(dh); % N/m2 ... pressure dif for transition
```

<sup>3</sup>Source Matlab files can be found on [ricopic.one/dynamic\\_systems\\_ii/source](http://ricopic.one/dynamic_systems_ii/source) as [ricopic.one/dynamic\\_systems\\_ii/source/nonlinear\\_tank\\_example.m](http://ricopic.one/dynamic_systems_ii/source/nonlinear_tank_example.m) and [ricopic.one/dynamic\\_systems\\_ii/source/nonlinear\\_fluid\\_state.m](http://ricopic.one/dynamic_systems_ii/source/nonlinear_fluid_state.m).





**Figure fluid.2:** nonlinear resistance versus tank water level.

### Nonlinear resistance

Now, let's define the variable resistance function  $R_{\text{fun}}(R(P_C))$ . We define the anonymous function via the two "limiting" resistances  $R_0$  ( $R_0$ ) and  $R_{\text{inf}}$  ( $R_\infty$ ).

```
R_inf = 1e-1; % N/m^2 / m^3/s ... resistance with full cap
R_0 = 1e2; % ... resistance with empty capacitor
R_fun = @(P) (R_0-R_inf)/2*(1-tanh(5/dP*(P-P_t)))+R_inf;
```

Let's take a moment to plot this function. See [Figure fluid.2](#) for the results. This is a reasonable approximation of a valve that allows no flow until the capacitor fluid height reaches a threshold, then allows a significant amount of flow.

```
h_half = linspace(0,h_t-dh,50);
h_a = [... % heights to plot
        h_half(1:end-1),...
        linspace(h_half(end),h_max,50)...
        ];
P_a = P_fun(h_a); % N/m^2 ... pressures to plot
```

*Numerical solution*

The numerical ODE solver we'll use (`ode23t`) requires we define the first-order system of differential equations from Equation 6. This is done by writing a function file `nonlinear_fluid_state.m` that the function return the time derivative of the state vector  $x_a(x)$  at a given time.

```
function dx = nonlinear_fluid_state(t,x,u_fun,R_fun)
global C L
% x(1) is P_C
% x(2) is Q_L
% R_fun is the nonlinear resistance

% call input function at this time step
Q_s = u_fun(t);

% compute nonlinear resistance at this time step
R = R_fun(x(1));

dx = zeros(2,1); % a column vector
dx(1) = (Q_s - x(2))/C; % d P_C/dt
dx(2) = (x(1) - x(2)*R)/L; % d Q_L/dt
end
```

We also pass it the nonlinear resistance function `R_fun` and the input function `Q_s_fun`. Let's model an offset sinusoidal input flowrate, defined as an anonymous function as follows.

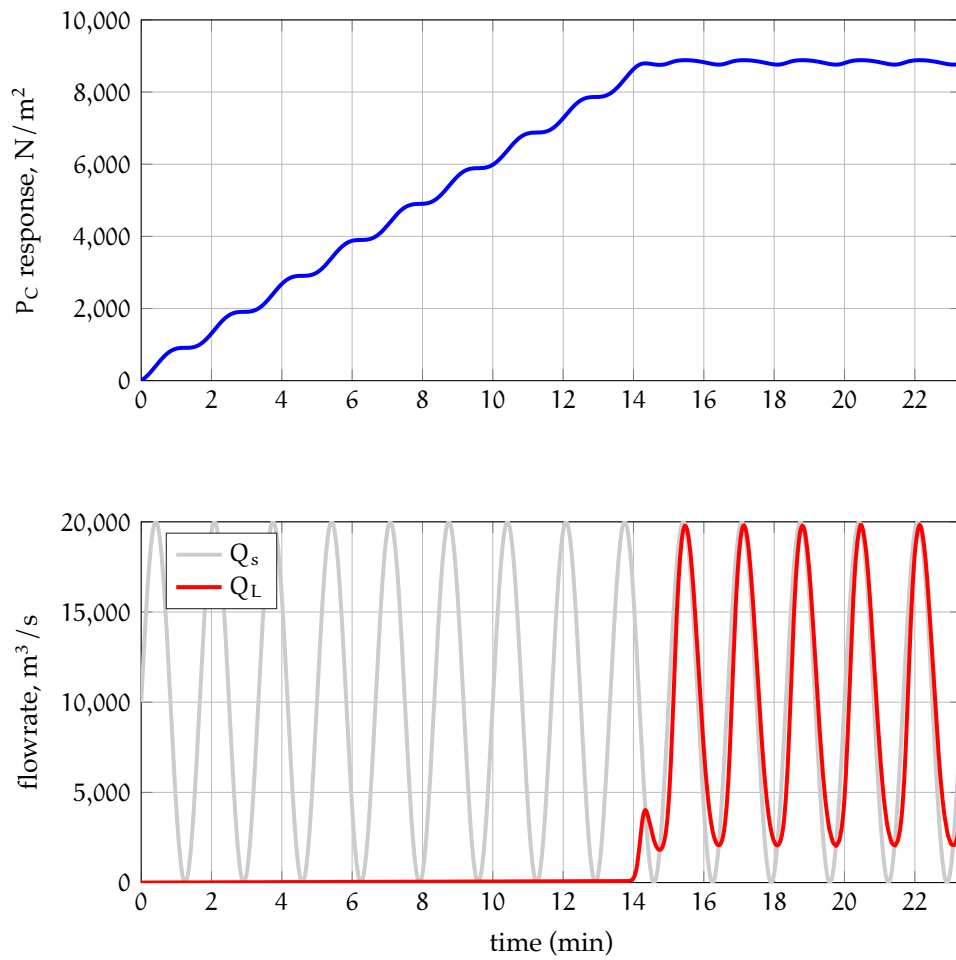
```
Q_s_fun = @(t) 1e4*(1+sin(2*pi/1e2*t));
```

We're ready to simulate! The time array and zero initial conditions are specified, then simulation commences. There are several Matlab ODE solver routines with the same or similar syntax. Many ODEs can be solved with the `ode45` function. However, this problem is what is called "stiff," which runs much better on the solver `ode23t`.

```
t_a = linspace(0,1.4e3,1e3);
x_0 = [0;0];
```

```
x_sol_struct = ode23t(...  
    @(t,x) nonlinear_fluid_state(t,x,Q_s_fun,R_fun),...  
    t_a,...  
    x_0...  
);  
x_sol = deval(x_sol_struct,t_a);
```

We plot the results in [Figure fluid.3](#). So the overflow is relatively inactive while the capacitor fills, until  $P_C$  achieves the pressure associated with a near-full capacitor. Then the flowrate suddenly increases rapidly due to the sudden drop in  $R(P_C)$ . Since the input is oscillating, the overflow pipe loses flowrate, then gains it again when the input flowrate increases enough to increase the capacitor pressure.



**Figure fluid.3:** state response to input  $Q_s$ .

## 16.4 sim.exe Exercises for Chapter 16 sim

### Exercise 16.1 freud

In [Exercise 14.1 nlin.](#), you derived a nonlinear state-space model for the RLC circuit of [Fig. exe.1](#), which includes a nonlinear capacitor, and linearized the state equation about an operating point. Use these results to perform the following analysis.

- Write a program to simulate the nonlinear state-space model for initial condition  $\mathbf{x}(0) = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$  and step input  $\mathbf{u}(t) = 5u_s(t)$ . Let  $R = 10 \Omega$ ,  $L = 1 \text{ mH}$ , and  $k = 10^{-6}$ . Try simulating for 1 ms.
- Add to the program the simulation of the *linearized* system for the same initial condition and input.
- Compare (by graphing) the nonlinear and linearized step responses. (Don't forget that  $\mathbf{x}^* \neq \mathbf{x}$ !)

### Exercise 16.2 kafka

Let the nonlinear state equation of a circuit like [Fig. exe.2](#), including a diode, be

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ &= \begin{bmatrix} \frac{1}{C} i_L \\ \frac{1}{L} (-V_{TH} \ln(i_L/I_s + 1) - R i_L + V_S - v_C) \end{bmatrix}. \end{aligned}$$

- Write a program to simulate the nonlinear state-space model for initial condition  $\mathbf{x}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$  and input  $\mathbf{u}(t) = 1 + 0.1 \cos(8000\pi t)$ . Let  $I_s = 10^{-12} \text{ A}$ ,  $V_{TH} = 25 \text{ mV}$ ,  $R = 10 \Omega$ ,  $L = 1 \text{ mH}$ , and  $C = 10 \mu\text{F}$ . Try simulating for 1 ms. *Hint*: the ode is stiff, so simulate with `ode23s`.

- b. Add to the program the simulation of the *linearized* system (with operating point  $\mathbf{x}_o = \begin{bmatrix} 0 & I_s \end{bmatrix}^\top$ ,  $u_o = 0$ ) with A and B matrices

$$A = \begin{bmatrix} 0 & 1/C \\ -1/L & -(V_{TH}/(2I_s) + R)/L \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 \\ 1/L \end{bmatrix}$$

for the same initial condition and input.

- c. Compare (by graphing) the nonlinear and linearized step responses. (Don't forget that  $\mathbf{x}^* \neq \mathbf{x}$ !)

### Exercise 16.3 hootenanny

Design a home rainwater catchment system and sprinkler distribution system. Most places, a surprising amount of water falls on a house's roof throughout a year. Capturing it for irrigation can save water costs and reduce the environmental impact of watering lawns, plants, and gardens. Design a home rainwater catchment and irrigation system. The design constraints are as follows.

1. It should be designed for Olympia, Washington rainfall, as described in [Table exe.1](#).
2. For a house, large tanks are unsightly. Instead, use a series of connected barrels.

After discussions with the customer, the following design requirements for the system are identified.

1. It should be capable of distributing one inch of water per unit area June through September, even during drought conditions, during which there is half the average rainfall in the months March through September (see [Table exe.1](#)).
2. The roof area for collection is 400 square ft.
3. The lawn area for distribution is 600 square ft.
4. It should be low-maintenance.

5. The distribution system should be capable of being “blown out” during winter months *or* it must be designed to handle sudden dips from 33 down to 22 deg F for up to two days.<sup>4</sup>
6. When tanks are full, it should be able to gracefully dump excess water. If possible, designing it to refresh itself by dumping old water for new water is desired.
7. It should be able to handle a heavy rain of 1 inch per hour via an overflow mechanism, but be able to handle a moderate rain of 0.2 inches per hour without requiring overflow (unless the tanks are full).
8. It should be designed to be fed from typical house rain gutter downspouts.
9. Distribution should be automated.
10. Energy efficiency is desired. If possible, using tanks’ potential energy for distribution is desired. In this case, unconventional distribution networks are allowable (e.g. “drip” systems without conventional sprinkler heads that require high pressure). However, the distribution hardware should not be custom-designed.
11. Commercially available parts are desired. Minimize the number of custom parts (zero is best).

The focus of the design problem is the sizing of the pipes, barrels, and mechanisms based on a dynamic system analysis.<sup>5</sup>

It is highly recommended that you use the following Fourier Series fit to the Olympia drought rainfall data, presented as trigonometric series coefficient vectors *a* and *b* for easy definition in Matlab.<sup>6</sup>

```
w = 0.5236; % fundamental frequency
a0 = 3.579; % dc offset
a(1) = 4.144;
b(1) = 0.6244;
a(2) = 1.332;
b(2) = 0.07578;
```

<sup>4</sup>A potential way to mitigate freezing is keeping the water in motion. Care must be taken not to create inadvertent ice skating rinks.

<sup>5</sup>A design that is not informed by a thoroughly presented system model will receive no credit.

<sup>6</sup>The fit is an 8-term Fourier series fit performed via Matlab’s `fit` function.

**Table exe.1:** mean monthly rainfall data and corresponding “drought conditions” for Olympia, Washington, USA (NOAA, 2017).

month	mean precip. (in)	drought precip. (in)
January	8.51	8.51
February	5.82	5.82
March	4.85	2.43
April	3.11	1.55
May	1.84	0.92
June	1.42	0.71
July	0.67	0.34
August	1.31	0.65
September	2.36	1.18
October	4.66	4.66
November	7.66	7.66
December	8.52	8.52

```

a(3) = -0.07667;
b(3) = 0.03167;
a(4) = -0.2469;
b(4) = 0.0004836;
a(5) = -0.09448;
b(5) = 0.01735;
a(6) = 0.07417;
b(6) = -2.131e-06;
a(7) = -0.06748;
b(7) = -0.0124;
a(8) = -0.1235;
b(8) = -0.0002381;

```

A system model response to this input can be used to determine the system parameters, such as the number of barrels required. Do not forget to include the effect of distribution, which can be modeled as a *negative* source.

Although we have the tools to perform the analysis analytically, it is highly recommended that a Matlab simulation is developed using `ss` to define the system and `lsim` to simulate the response. A frequency response analysis using `bode` may also prove useful. It may be possible to simply iteratively tweak design parameters until the simulation meets the requirements.



A thorough report is required. It is highly recommended that LaTeX is used. Thorough analysis, results, and design is required. All sizing and specific parts are required. Either an analytic or a numerical (simulation) demonstration of the design's fulfillment of the requirements is required.

## **Part VII**

# **Appendices**

**A**

## 01.1 math.quad Quadratic forms

The solution to equations of the form  $ax^2 + bx + c = 0$  is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \quad (1)$$

### Completing the square

This is accomplished by re-writing the quadratic formula in the form of the left-hand-side (LHS) of this equality, which describes factorization

$$x^2 + 2xh + h^2 = (x + h)^2. \quad (2)$$

## 01.2 math.trig Trigonometry

### Triangle identities

With reference to the below figure, the *law of sines* is

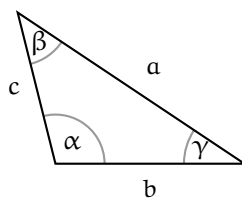
$$\frac{\sin \alpha}{a} = \frac{\sin \beta}{b} = \frac{\sin \gamma}{c} \quad (1)$$

and the *law of cosines* is

$$c^2 = a^2 + b^2 - 2ab \cos \gamma \quad (2a)$$

$$b^2 = a^2 + c^2 - 2ac \cos \beta \quad (2b)$$

$$a^2 = c^2 + b^2 - 2cb \cos \alpha \quad (2c)$$



### Reciprocal identities

$$\csc u = \frac{1}{\sin u} \quad (3a)$$

$$\sec u = \frac{1}{\cos u} \quad (3b)$$

$$\cot u = \frac{1}{\tan u} \quad (3c)$$

### Pythagorean identities

$$1 = \sin^2 u + \cos^2 u \quad (4a)$$

$$\sec^2 u = 1 + \tan^2 u \quad (4b)$$

$$\csc^2 u = 1 + \cot^2 u \quad (4c)$$

**Co-function identities**

$$\sin\left(\frac{\pi}{2} - u\right) = \cos u \quad (5a)$$

$$\cos\left(\frac{\pi}{2} - u\right) = \sin u \quad (5b)$$

$$\tan\left(\frac{\pi}{2} - u\right) = \cot u \quad (5c)$$

$$\csc\left(\frac{\pi}{2} - u\right) = \sec u \quad (5d)$$

$$\sec\left(\frac{\pi}{2} - u\right) = \csc u \quad (5e)$$

$$\cot\left(\frac{\pi}{2} - u\right) = \tan u \quad (5f)$$

**Even-odd identities**

$$\sin(-u) = -\sin u \quad (6a)$$

$$\cos(-u) = \cos u \quad (6b)$$

$$\tan(-u) = -\tan u \quad (6c)$$

**Sum-difference formulas (AM or lock-in)**

$$\sin(u \pm v) = \sin u \cos v \pm \cos u \sin v \quad (7a)$$

$$\cos(u \pm v) = \cos u \cos v \mp \sin u \sin v \quad (7b)$$

$$\tan(u \pm v) = \frac{\tan u \pm \tan v}{1 \mp \tan u \tan v} \quad (7c)$$

**Double angle formulas**

$$\sin(2u) = 2 \sin u \cos u \quad (8a)$$

$$\cos(2u) = \cos^2 u - \sin^2 u \quad (8b)$$

$$= 2 \cos^2 u - 1 \quad (8c)$$

$$= 1 - 2 \sin^2 u \quad (8d)$$

$$\tan(2u) = \frac{2 \tan u}{1 - \tan^2 u} \quad (8e)$$

**Power-reducing or half-angle formulas**

$$\sin^2 u = \frac{1 - \cos(2u)}{2} \quad (9a)$$

$$\cos^2 u = \frac{1 + \cos(2u)}{2} \quad (9b)$$

$$\tan^2 u = \frac{1 - \cos(2u)}{1 + \cos(2u)} \quad (9c)$$

**Sum-to-product formulas**

$$\sin u + \sin v = 2 \sin \frac{u+v}{2} \cos \frac{u-v}{2} \quad (10a)$$

$$\sin u - \sin v = 2 \cos \frac{u+v}{2} \sin \frac{u-v}{2} \quad (10b)$$

$$\cos u + \cos v = 2 \cos \frac{u+v}{2} \cos \frac{u-v}{2} \quad (10c)$$

$$\cos u - \cos v = -2 \sin \frac{u+v}{2} \sin \frac{u-v}{2} \quad (10d)$$

**Product-to-sum formulas**

$$\sin u \sin v = \frac{1}{2} [\cos(u-v) - \cos(u+v)] \quad (11a)$$

$$\cos u \cos v = \frac{1}{2} [\cos(u-v) + \cos(u+v)] \quad (11b)$$

$$\sin u \cos v = \frac{1}{2} [\sin(u+v) + \sin(u-v)] \quad (11c)$$

$$\cos u \sin v = \frac{1}{2} [\sin(u+v) - \sin(u-v)] \quad (11d)$$

**Two-to-one or harmonic amplitude formulas**

$$a \cos u + b \sin u = c \sin(u + \phi) \quad (12a)$$

$$= c \cos(u + \psi) \text{ where} \quad (12b)$$

$$c = \sqrt{a^2 + b^2} \quad (12c)$$

$$\phi = \arctan \frac{a}{b} \quad (12d)$$

$$\psi = -\arctan \frac{b}{a} \quad (12e)$$

## 01.3 math.matrix Matrix inverses

This is a guide to inverting  $1 \times 1$ ,  $2 \times 2$ , and  $n \times n$  matrices.

Let  $A$  be the  $1 \times 1$  matrix

$$A = [a].$$

The inverse is simply the reciprocal:

$$A^{-1} = [1/a].$$

Let  $B$  be the  $2 \times 2$  matrix

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

It can be shown that the inverse follows a simple pattern:

$$\begin{aligned} B^{-1} &= \frac{1}{\det B} \begin{bmatrix} b_{22} & -b_{12} \\ -b_{21} & b_{11} \end{bmatrix} \\ &= \frac{1}{b_{11}b_{22} - b_{12}b_{21}} \begin{bmatrix} b_{22} & -b_{12} \\ -b_{21} & b_{11} \end{bmatrix}. \end{aligned}$$

Let  $C$  be an  $n \times n$  matrix. It can be shown that its inverse is

$$C^{-1} = \frac{1}{\det C} \text{adj } C,$$

where  $\text{adj}$  is the **adjoint** of  $C$ .



## 01.4 math.lap Laplace transforms

The definition of the one-side Laplace and inverse Laplace transforms follow.

### Definition A.1: Laplace transforms (one-sided)

Laplace transform  $\mathcal{L}$ :

$$\mathcal{L}(y(t)) = Y(s) = \int_0^{\infty} y(t)e^{-st} dt. \quad (1)$$

Inverse Laplace transform  $\mathcal{L}^{-1}$ :

$$\mathcal{L}^{-1}(Y(s)) = y(t) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} Y(s)e^{st} ds. \quad (2)$$

See [Table lap.1](#) for a list of properties and common transforms.

**B**

## 02.1 adv.eig Systems with repeated eigenvalues

This topic is fully treated by Brogan (1991, p 250), but not by Rowell and Wormley (1997). Every  $n \times n$  matrix has  $n$  eigenvalues, and for each distinct eigenvalue  $\lambda_i$ , a linear independent eigenvector  $\mathbf{m}_i$  exists. For every eigenvalue  $\lambda_i$  repeated  $\mu_i$  times (termed *algebraic multiplicity* of  $\lambda_i$ ), any number  $q_i$  (termed *geometric multiplicity* or *degeneracy* of  $\lambda_i$ ) up to and including  $\mu_i$  of independent eigenvectors may exist:  $1 \leq q_i \leq \mu_i$ .  $q_i$  is equal to the dimension of the null space of  $A - \lambda_i \mathbf{I}$ ,

$$q_i = n - \text{rank}(A - \lambda_i \mathbf{I}). \quad (1)$$

This gives rise to the three cases that follow.

**full degeneracy** When  $q_i = \mu_i$ , the eigenvalue problem has  $q_i = \mu_i$  independent solutions for  $\mathbf{m}_i$ . So, even though there were not  $n$  distinct eigenvalues,  $n$  distinct eigenvectors still exist and we can diagonalize or decouple the system as before.

**simple degeneracy** When  $q_i = 1$ , the eigenvalue problem has  $q_i = 1$  independent solutions for  $\mathbf{m}_i$ . We would still like to construct a basis set of  $n$  independent vectors, but they can no longer be eigenvectors, and we will no longer be able to fully diagonalize or decouple the system. There are multiple ways of doing this (e.g. Gram-Schmidt), but the typical and most nearly diagonal way is to construct  $\mu_i - q_i$  *generalized eigenvectors* (here also called  $\mathbf{m}_i$ ), which will be included in the modal matrix  $M$  along with the eigenvectors. The generalized eigenvectors are found by solving the usual eigenvalue/vector problem for the first eigenvector  $\mathbf{m}_i^1$  corresponding to  $\lambda_i$ , then solving it again with the following equations to find the generalized eigenvectors

$$\begin{aligned} (A - \lambda_i)\mathbf{m}_i^2 &= \mathbf{m}_i^1 \\ (A - \lambda_i)\mathbf{m}_i^3 &= \mathbf{m}_i^2 \\ &\vdots \end{aligned}$$

This forms the modal matrix  $M$ . The block-diagonal *Jordan form* matrix  $J$ , analogous to the diagonal  $\Lambda$  is

$$J = M^{-1}AM, \quad (2)$$

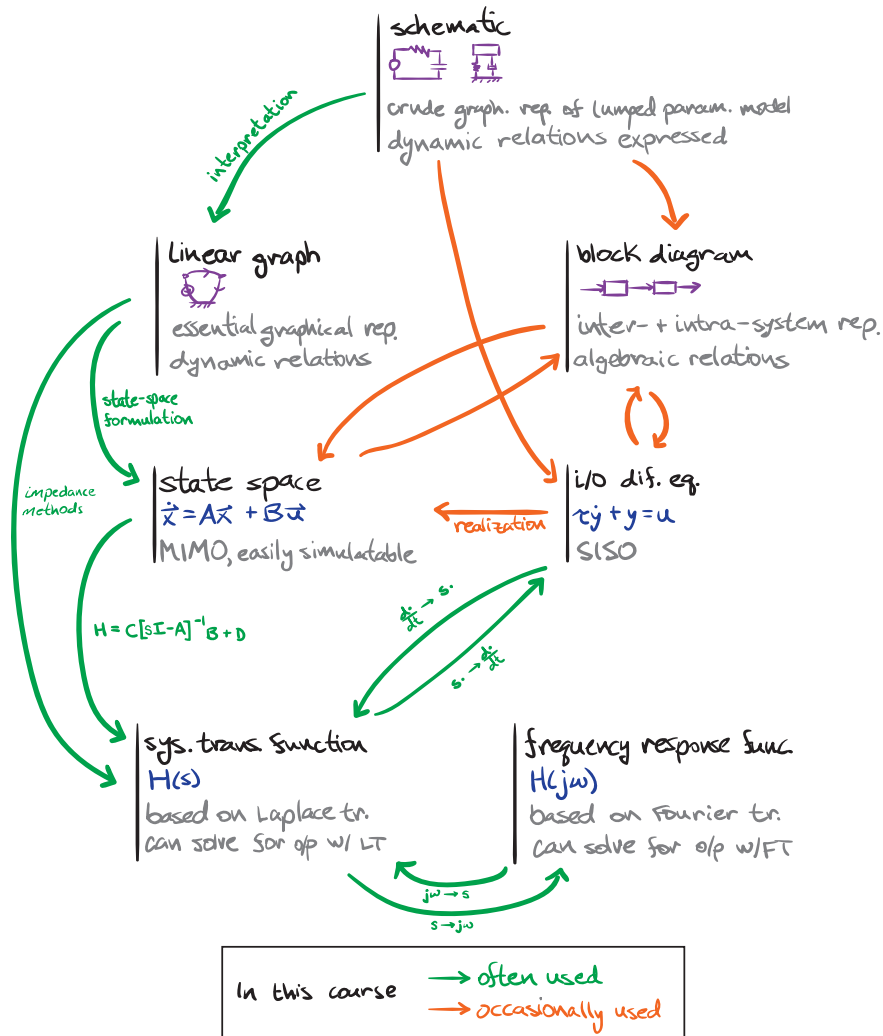
which gives the most-decoupled state transition matrix

$$\Phi(t) = Me^{Jt}M^{-1}. \quad (3)$$

**general degeneracy** If  $1 < q_i < \mu_i$ , the preceding method applies, but it may be ambiguous as to which eigenvector the generalized eigenvectors correspond (or how many for each). This can be approached by trial and error or a systematic method presented by Brogan (1991, p 255).

C

# 03.1 sum.sysrep Summary of system representations



**Figure sysrep.1:** relations among system representations.

## 03.2 sum.els Summary of one-port elements

The one table to rule them all, [Table els.1](#).

**Table els.1:** parameters, elemental equations, and impedances of one-port elements for generalized, mechanical, electrical, fluid, and thermal systems.

		generalized	mechanical translation	mechanical rotation	electrical	fluid	thermal
<b>variables</b>	<i>across</i>	$\mathcal{V}$	velocity $v$	angular vel. $\Omega$	voltage $v$	pressure $P$	temp. $T$
	<i>through</i>	$\mathcal{F}$	force $f$	torque $T$	current $i$	vol. fr. $Q$	heat fr. $q$
<b>A-type</b>	<i>capacitor</i>	capacitor	mass	mom. inertia	capacitor	capacitor	capacitor
	<i>capacitance</i>	$C$	$m$	$J$	$C$	$C$	$C$
	<i>elem. eq.</i>	$\frac{d\mathcal{V}_C}{dt} = \frac{1}{C}\mathcal{F}_C$	$\frac{dv_m}{dt} = \frac{1}{m}f_m$	$\frac{d\Omega_J}{dt} = \frac{1}{J}T_J$	$\frac{dv_C}{dt} = \frac{1}{C}i_C$	$\frac{dP_C}{dt} = \frac{1}{C}Q_C$	$\frac{dT_C}{dt} = \frac{1}{C}q_C$
	<i>impedance</i>	$\frac{1}{Cs}$	$\frac{1}{ms}$	$\frac{1}{Js}$	$\frac{1}{Cs}$	$\frac{1}{Cs}$	$\frac{1}{Cs}$
<b>T-type</b>	<i>inductor</i>	inductor	spring	rot. spring	inductor	inertance	
	<i>inductance</i>	$L$	$1/k$	$1/k$	$L$	$I$	
	<i>elem. eq.</i>	$\frac{d\mathcal{F}_L}{dt} = \frac{1}{L}\mathcal{V}_L$	$\frac{df_k}{dt} = kv_k$	$\frac{dT_k}{dt} = k\Omega_k$	$\frac{di_L}{dt} = \frac{1}{L}v_L$	$\frac{dQ_I}{dt} = \frac{1}{I}P_I$	
	<i>impedance</i>	$Ls$	$s/k$	$s/k$	$Ls$	$Is$	
<b>D-type</b>	<i>resistor</i>	resistor	damper	rot. damper	resistor	resistor	resistor
	<i>resistance</i>	$R$	$1/B$	$1/B$	$R$	$R$	$R$
	<i>elem. eq.</i>	$\mathcal{V}_R = \mathcal{F}_R R$	$v_B = f_B/B$	$\Omega_B = T_B/B$	$v_R = i_R R$	$P_R = Q_R R$	$T_R = q_R R$
	<i>impedance</i>	$R$	$1/B$	$1/B$	$R$	$R$	$R$

### 03.3 sum.lap Laplace transforms

Table lap.1 is a table with functions of time  $f(t)$  on the left and corresponding Laplace transforms  $L(s)$  on the right. Where applicable,  $s = \sigma + j\omega$  is the Laplace transform variable,  $T$  is the time-domain period,  $\omega_0 2\pi/T$  is the corresponding angular frequency,  $j = \sqrt{-1}$ ,  $a \in \mathbb{R}^+$ , and  $b, t_0 \in \mathbb{R}$  are constants.

**Table lap.1:** Laplace transform identities.

function of time $t$	function of Laplace $s$
$a_1 f_1(t) + a_2 f_2(t)$	$a_1 F_1(s) + a_2 F_2(s)$
$f(t - t_0)$	$F(s)e^{-t_0 s}$
$f'(t)$	$sF(s) - f(0)$
$\frac{d^n f(t)}{dt^n}$	$s^n F(s) + s^{(n-1)} f(0) + s^{(n-2)} f'(0) + \dots + f^{(n-1)}(0)$
$\int_0^t f(\tau) d\tau$	$\frac{1}{s} F(s)$
$tf(t)$	$-F'(s)$
$f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t - \tau) d\tau$	$F_1(s) F_2(s)$
$\delta(t)$	1
$u_s(t)$	$1/s$
$u_r(t)$	$1/s^2$
$t^{n-1}/(n-1)!$	$1/s^n$



$e^{-at}$	$\frac{1}{s+a}$
$te^{-at}$	$\frac{1}{(s+a)^2}$
$\frac{1}{(n-1)!}t^{n-1}e^{-at}$	$\frac{1}{(s+a)^n}$
$\frac{1}{a-b}(e^{at} - e^{bt})$	$\frac{1}{(s-a)(s-b)} \quad (a \neq b)$
$\frac{1}{a-b}(ae^{at} - be^{bt})$	$\frac{s}{(s-a)(s-b)} \quad (a \neq b)$
$\sin \omega t$	$\frac{\omega}{s^2 + \omega^2}$
$\cos \omega t$	$\frac{s}{s^2 + \omega^2}$
$e^{at} \sin \omega t$	$\frac{\omega}{(s-a)^2 + \omega^2}$
$e^{at} \cos \omega t$	$\frac{s-a}{(s-a)^2 + \omega^2}$

---

## 03.4 sum.ft Fourier transforms

Table ft.1 is a table with functions of time  $f(t)$  on the left and corresponding Fourier transforms  $F(\omega)$  on the right. Where applicable,  $T$  is the time-domain period,  $\omega_0 2\pi/T$  is the corresponding angular frequency,  $j = \sqrt{-1}$ ,  $a \in \mathbb{R}^+$ , and  $b, t_0 \in \mathbb{R}$  are constants. Furthermore,  $f_e$  and  $f_o$  are even and odd functions of time, respectively, and it can be shown that any function  $f$  can be written as the sum  $f(t) = f_e(t) + f_o(t)$ . (Hsu, 1967)

**Table ft.1:** Fourier transform identities.

function of time $t$	function of frequency $\omega$
$a_1 f_1(t) + a_2 f_2(t)$	$a_1 F_1(\omega) + a_2 F_2(\omega)$
$f(at)$	$\frac{1}{ a } F(\omega/a)$
$f(-t)$	$F(-\omega)$
$f(t - t_0)$	$F(\omega)e^{-j\omega t_0}$
$f(t) \cos \omega_0 t$	$\frac{1}{2} F(\omega - \omega_0) + \frac{1}{2} F(\omega + \omega_0)$
$f(t) \sin \omega_0 t$	$\frac{1}{j2} F(\omega - \omega_0) - \frac{1}{j2} F(\omega + \omega_0)$
$f_e(t)$	$\text{Re } F(\omega)$
$f_o(t)$	$j \text{Im } F(\omega)$
$F(t)$	$2\pi f(-\omega)$
$f'(t)$	$j\omega F(\omega)$

$\frac{d^n f(t)}{dt^n}$	$(j\omega)^n F(\omega)$
$\int_{-\infty}^t f(\tau) d\tau$	$\frac{1}{j\omega} F(\omega) + \pi F(0)\delta(\omega)$
$-jtf(t)$	$F'(\omega)$
$(-jt)^n f(t)$	$\frac{d^n F(\omega)}{d\omega^n}$
$f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t - \tau) d\tau$	$F_1(\omega) F_2(\omega)$
$f_1(t) f_2(t)$	$\frac{1}{2\pi} F_1(\omega) * F_2(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_1(\alpha) F_2(\omega - \alpha) d\alpha$
$e^{-at} u_s(t)$	$\frac{1}{j\omega + a}$
$e^{-a t }$	$\frac{2a}{a^2 + \omega^2}$
$e^{-at^2}$	$\sqrt{\pi/a} e^{-\omega^2/(4a)}$
1 for $ t  < a/2$ , else 0	$\frac{a \sin(a\omega/2)}{a\omega/2}$
$te^{-at} u_s(t)$	$\frac{1}{(a + j\omega)^2}$
$\frac{t^{n-1}}{(n-1)!} e^{-at} u_s(t)$	$\frac{1}{(a + j\omega)^n}$
$\frac{1}{a^2 + t^2}$	$\frac{\pi}{a} e^{-a \omega }$
$\delta(t)$	1
$\delta(t - t_0)$	$e^{-j\omega t_0}$
$u_s(t)$	$\pi\delta(\omega) + \frac{1}{j\omega}$

$u_s(t - t_0)$	$\pi\delta(\omega) + \frac{1}{j\omega}e^{-j\omega t_0}$
1	$2\pi\delta(\omega)$
t	$2\pi j\delta'(\omega)$
$t^n$	$2\pi j^n \frac{d^n \delta(\omega)}{d\omega^n}$
$e^{j\omega_0 t}$	$2\pi\delta(\omega - \omega_0)$
$\cos \omega_0 t$	$\pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0)$
$\sin \omega_0 t$	$-j\pi\delta(\omega - \omega_0) + j\pi\delta(\omega + \omega_0)$
$u_s(t) \cos \omega_0 t$	$\frac{j\omega}{\omega_0^2 - \omega^2} + \frac{\pi}{2}\delta(\omega - \omega_0) + \frac{\pi}{2}\delta(\omega + \omega_0)$
$u_s(t) \sin \omega_0 t$	$\frac{\omega_0}{\omega_0^2 - \omega^2} + \frac{\pi}{2j}\delta(\omega - \omega_0) - \frac{\pi}{2j}\delta(\omega + \omega_0)$
$tu_s(t)$	$j\pi\delta'(\omega) - 1/\omega^2$
1/t	$\pi j - 2\pi j u_s(\omega)$
1/t <sup>n</sup>	$\frac{(-j\omega)^{n-1}}{(n-1)!} (\pi j - 2\pi j u_s(\omega))$
sgn t	$\frac{2}{j\omega}$
$\sum_{n=-\infty}^{\infty} \delta(t - nT)$	$\omega_0 \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0)$

---

**D**

## 04.1 com.euler Euler's formulas

**Euler's formula** is our bridge back-and forth between trigonometric forms ( $\cos \theta$  and  $\sin \theta$ ) and complex exponential form ( $e^{j\theta}$ ):

$$e^{j\theta} = \cos \theta + j \sin \theta. \quad (1)$$

Here are a few useful identities implied by Euler's formula.

$$e^{-j\theta} = \cos \theta - j \sin \theta \quad (2a)$$

$$\cos \theta = \operatorname{Re}(e^{j\theta}) \quad (2b)$$

$$= \frac{1}{2}(e^{j\theta} + e^{-j\theta}) \quad (2c)$$

$$\sin \theta = \operatorname{Im}(e^{j\theta}) \quad (2d)$$

$$= \frac{1}{j2}(e^{j\theta} - e^{-j\theta}). \quad (2e)$$

- 
- Authority, Tennessee Valley **and** Tomia (2018). *Hydroelectric dam*—*Wikipedia, The Free Encyclopedia*. [Online; accessed 13-February-2018].
- Baldursson, Stefán (2005). ?BLDC Motor Modelling and Control – A Matlab®/Simulink®Implementation? mathesis. Chalmers University.
- Booton, Richard C. **and** Simon Ramo (july 1984). ?The development of systems engineering? **in** *IEEE Transactions on Aerospace and Electronic Systems: AES-20*, **pages** 306–9.
- Brogan, William L (1991). *Modern Control Theory*. Third. Prentice Hall.
- Choukchou-Braham, A. **and** others (2013). *Analysis and Control of Underactuated Mechanical Systems*. SpringerLink : B ucher. Springer International Publishing. ISBN: 9783319026367.
- Devine, Cameron N. **and** Rico A.R. Picone (2018). *Statum*.  
<https://github.com/CameronDevine/Statum>.
- Dyke, P.P.G. (2014). *An Introduction to Laplace Transforms and Fourier Series*. 2 **edition**. Springer Undergraduate Mathematics Series. Springer. ISBN: 9781447163954.
- Hsu, H.P. (1967). *Fourier Analysis*. Simon & Schuster. ISBN: 9780671270377.
- Mathews, John H. **and** Russell W. Howell (2012). *Complex Analysis for Mathematics and Engineering*. 6 **edition**. Jones **and** Bartlett Publishers. ISBN: 9781449604455.
- Nise, N.S. (2015). *Control Systems Engineering, 7th Edition*. Wiley. ISBN: 9781118800829.
- NOAA (**august** 2017). *Mohtly Average Precipitation 1951-2008 Olympia Regional Airport*—*NOAA Station*.

- Partington, Jonathan R. (2004). *Linear operators and linear systems: An analytical approach to control theory*. London Mathematical Society Student Texts. CUP. ISBN: 9780521546195.
- Rowell, Derek **and** David N. Wormley (1997). *System Dynamics: An Introduction*. Prentice Hall.
- Strogatz, S.H. **and** M. Dichter (2016). *Nonlinear Dynamics and Chaos*. Second. Studies in Nonlinearity. Avalon Publishing. ISBN: 9780813350844.